

TOWARD ROBUST VIDEO EVENT DETECTION AND RETRIEVAL UNDER
ADVERSARIAL CONSTRAINTS

Yi Xu

A dissertation submitted to the faculty of the University of North Carolina at Chapel Hill in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the Department of Computer Science.

Chapel Hill
2016

Approved by:

Jan-Michael Frahm

Fabian Monronse

Enrique Dunn

Tamara L. Berg

David Crandall

©2016
Yi Xu
ALL RIGHTS RESERVED

ABSTRACT

Yi Xu: Toward Robust Video Event Detection and Retrieval Under Adversarial Constraints
(Under the direction of Jan-Michael Frahm and Fabian Monronse)

The continuous stream of videos that are uploaded and shared on the Internet has been leveraged by computer vision researchers for a myriad of detection and retrieval tasks, including gesture detection, copy detection, face authentication, *etc.* However, the existing state-of-the-art event detection and retrieval techniques fail to deal with several real-world challenges (*e.g.*, low resolution, low brightness and noise) under adversary constraints. This dissertation focuses on these challenges in realistic scenarios and demonstrates practical methods to address the problem of robustness and efficiency within video event detection and retrieval systems in five application settings (namely, CAPTCHA decoding, face liveness detection, reconstructing typed input on mobile devices, video confirmation attack, and content-based copy detection).

Specifically, for CAPTCHA decoding, I propose an automated approach which can decode moving-image object recognition (MIOR) CAPTCHAs faster than humans. I showed that not only are there inherent weaknesses in current MIOR CAPTCHA designs, but that several obvious countermeasures (*e.g.*, extending the length of the codeword) are not viable. More importantly, my work highlights the fact that the choice of underlying hard problem selected by the designers of a leading commercial solution falls into a solvable subclass of computer vision problems.

For face liveness detection, I introduce a novel approach to bypass modern face authentication systems. More specifically, by leveraging a handful of pictures of the target user taken from social media, I show how to create realistic, textured, 3D facial models that undermine the security of widely used face authentication solutions. My framework makes use of virtual reality (VR) systems, incorporating along the way the ability to perform animations (*e.g.*, raising an eyebrow or smiling) of the facial model, in order to trick liveness detectors into believing that the 3D model is a real

human face. I demonstrate that such VR-based spoofing attacks constitute a fundamentally new class of attacks that point to a serious weaknesses in camera-based authentication systems.

For reconstructing typed input on mobile devices, I proposed a method that successfully transcribes the text typed on a keyboard by exploiting video of the user typing, even from significant distances and from repeated reflections. This feat allows us to reconstruct typed input from the image of a mobile phone’s screen on a user’s eyeball as reflected through a nearby mirror, extending the privacy threat to include situations where the adversary is located around a corner from the user.

To assess the viability of a video confirmation attack, I explored a technique that exploits the emanations of changes in light to reveal the programs being watched. I leverage the key insight that the observable emanations of a display (e.g., a TV or monitor) during presentation of the viewing content induces a distinctive flicker pattern that can be exploited by an adversary. My proposed approach works successfully in a number of practical scenarios, including (but not limited to) observations of light effusions through the windows, on the back wall, or off the victim’s face. My empirical results show that I can successfully confirm hypotheses while capturing short recordings (typically less than 4 minutes long) of the changes in brightness from the victim’s display from a distance of 70 meters.

Lastly, for content-based copy detection, I take advantage of a new temporal feature to index a reference library in a manner that is robust to the popular spatial and temporal transformations in pirated videos. My technique narrows the detection gap in the important area of temporal transformations applied by would-be pirates. My large-scale evaluation on real-world data shows that I can successfully detect infringing content from movies and sports clips with 90.0% precision at a 71.1% recall rate, and can achieve that accuracy at an average time expense of merely 5.3 seconds, outperforming the state of the art by an order of magnitude.

ACKNOWLEDGEMENTS

I would like to thank my advisors, Jan-Michael Frahm and Fabian Monrose, for their input and guidance.

I would also like to thank my committee members, Tamara L. Berg, Enrique Dunn, and David Crandall, for their feedback and advice.

Additionally, I would like to thank my labmates, as their company and discussion made my time more fruitful and enjoyable: Akash Bapat, Sangwoo Cho, Pierre Fite-Georgel, Jeremy Graham, Jared Heinly, Yunchao Gong, Rohit Gupta, Shubham Gupta, Xufeng Han, Junpyo Hong, Yi-Hung Jen, Dinghuang Ji, Alex Keng, Hadi Kiapour, Hyo Jin Kim, Wei Liu, Jie Lu, Vicente Ordóñez-Román, David Perra, True Price, Rahul Raguram, Patrick Reynolds, Johannes Schönberger, Meng Tan, Joseph Tighe, Sirion Vittayakorn, Ke Wang, Yilin Wang, Hongsheng Yang, and Enliang Zheng.

I would like to thank my parents, who encouraged me to explore, create, think, and learn.

TABLE OF CONTENTS

LIST OF TABLES	xi
LIST OF FIGURES	xiii
LIST OF ABBREVIATIONS	xviii
LIST OF SYMBOLS	xix
1 Introduction	1
1.1 Thesis Statement	3
1.2 List of Innovations	4
1.2.1 Decoding Codewords from Motion-based Captchas	4
1.2.2 Defeating Face Liveness Detection with Virtual Models	5
1.2.3 Reconstructing Typed Input from Repeated Reflections	6
1.2.4 Inferring TV Content from Light Effusions	7
1.2.5 Detecting Copies from Large Video Collections	8
2 Background and Related Work	10
2.1 Video Event Detection and Retrieval	10
2.1.1 Video Event Detection	10
2.1.2 Video Event Retrieval	12
2.2 The Security of Moving Object CAPTCHAs	13
2.2.1 Adversary Modeling	14
2.2.2 Related Work	16
2.2.3 Remaining Challenges	18
2.3 The Security of Face Liveness Detection	18

2.3.1	Adversary Modeling	19
2.3.2	Related Work	20
2.3.3	Remaining Challenges	21
2.3.3.1	Online Photos and Face Authentication	22
2.3.3.2	Defeating Facial Liveness Detection by Building Virtual Models From Public Photos.....	23
2.4	Compromising Optical Emanation.....	25
2.4.1	Adversary Modeling	27
2.4.1.1	Reconstructing Typed Input from Repeated Reflections	27
2.4.1.2	Inferring TV Content from Light Effusions	31
2.4.2	Most Related Works	35
2.4.2.1	Related Work in Security	36
2.4.2.2	Related Work in Recognition and Retrieval	38
2.4.3	Remaining Challenges	40
2.5	Content-Based Copy Detection (CBCD)	40
2.5.1	Adversary Modeling	41
2.5.1.1	Findings	43
2.5.1.2	Assumptions	45
2.5.2	Related Work	45
2.5.2.1	Spatial Features	46
2.5.2.2	Temporal Features.....	48
2.5.2.3	Indexing	48
2.5.2.4	Video Quality Assessment	49
2.5.2.5	Audio Matching.....	50
3	Decoding Codewords from Motion-based Captchas.....	51
3.1	Introduction	51
3.2	Our Automated Approach.....	53

3.2.1	A Naïve Attack	54
3.2.2	Exploiting Temporal Information	56
3.2.3	Codeword Extraction and Classification	62
3.3	Evaluation	64
3.3.1	Results	64
3.3.2	Mitigation	65
3.3.2.1	Results	69
3.3.2.2	Discussion	70
3.4	User study	72
3.4.1	Methodology	72
3.4.2	Data Collection	74
3.4.3	Analysis	74
3.5	Summary and Concluding Remarks	79
4	Defeating Face Liveness Detection with Virtual Models.....	81
4.1	Introduction	81
4.2	Our Approach	84
4.2.1	Facial Landmark Extraction	85
4.2.2	3D Model Reconstruction	85
4.2.3	Facial Texture Patching	88
4.2.4	Gaze Correction	90
4.2.5	Adding Facial Animations	92
4.2.6	Leveraging Virtual Reality	92
4.3	Evaluation	94
4.3.1	Evaluating System Robustness.....	99
4.3.1.1	Blurry, Grainy Pictures Still Say A Lot	99
4.3.1.2	A Little to the Left, a Little to the Right	100

4.3.1.3	For Want of a Selfie	101
4.3.2	Seeing Your Face <i>Is</i> Enough	102
4.4	Defense in Depth	103
5	Reconstructing Typed Input from Repeated Reflections	106
5.1	Introduction	106
5.2	Automated Transcription	108
5.3	Evaluation	120
5.3.1	Results	122
5.3.1.1	Password Guessing	126
5.3.1.2	Runtime Performance	129
5.4	Mitigations	129
5.5	Conclusions	130
6	Inferring TV Content from Light Effusions	131
6.1	Introduction	131
6.2	Overview	133
6.3	Automated Video Retrieval	134
6.3.1	Feature Extraction	134
6.3.2	Creating the Reference Library	136
6.3.3	Locating the Best Matching Sequences	136
6.4	Illuminati: Efficient Attacks using Compromising Effusions	139
6.4.1	Peak-feature Extraction	139
6.4.2	Efficient Searching	141
6.5	Evaluation	143
6.5.1	Lights On	147
6.5.2	Impact of Screen Size	148
6.5.3	Impact of Reference Library Size	149

6.5.4	As Seen From Outdoors	151
6.6	Mitigations.....	152
6.7	Conclusion and Future Work	154
7	Detecting Copies from Large Video Collections	157
7.1	Introduction.....	157
7.2	Approach	159
7.2.1	Derivative Feature Extraction	160
7.2.2	Local Feature Extraction	161
7.2.3	Indexing and Retrieval	162
7.2.4	Detecting Copied sub-sequences	163
7.3	Evaluation	164
7.4	Conclusion.....	171
8	Discussion	173
	APPENDIX.....	178
A	Parameters for video generation	178
B	Comments from User Study.....	178
C	Multi-Image Facial Model Estimation	178
D	Anisotropic Diffusion	180
E	Vanishing Points	181
F	Translation Alignment	182
	BIBLIOGRAPHY.....	184

LIST OF TABLES

2.1	RRED index score statistics for copy-reference pairs in the IACC and VCDB dataset, compared with the result from random videos	43
2.2	RRED index score statistics for copy-reference pairs in IACC and VCDB dataset, compared with result from random different videos	49
3.1	Reconstruction accuracy for various attacks.....	64
4.1	Summary of the face authentication systems evaluated. The second column lists how each system acquires training data for learning a user’s face, and the third column shows the number approximate number of installations or reviews each system has received according to (1) the Google Play Store, (2) the iTunes store, or (3) softpedia.com. BioID is a relatively new app and does not yet have customer reviews on iTunes.	96
4.2	Success rate for 5 face authentication systems using a model built from (second column) an image of the user taken in an indoor environment and (third and fourth columns) images obtained on users’ social media accounts. The fourth column shows the average number of attempts needed before successfully spoofing the target user.	98
4.3	Number of testing samples classified as real users. Values in the first column represent true positive rates, and the second and third columns represent false positives. Each row shows the classification results after training on the classes in the first column. The results were averaged over four trials.	103
5.1	Specifications of the cameras used in my evaluation.	120
5.2	Example reconstructions. [C1]: Using Canon 60D DSLR(\$700) with 400mm Lens(\$1340). [C2]: Using Canon VIXIA HG21 Camcorder(\$1000).....	122
5.3	Expected number of guesses required to identify user passwords given a rank-ing, by decreasing confidence, of candidates.	128
5.4	Runtime performance of my approach.	129
6.1	Retrieval success rate with random start point.	145
6.2	Worst case capture length with different illumination settings.	147
6.3	Worst case capture length with different screen size.	149
7.1	Recall rate for different temporal scaling	168

B1	Sample participant comments for each variant	179
----	--	-----

LIST OF FIGURES

2.1	Copy-reference pairs in the VCDB dataset	42
2.2	Viewership statistics for the videos in the VCDB dataset	44
3.1	Example moving-image object recognition (MIOR) CAPTCHAs from Nu-Captcha (see http://nucaptcha.com/demo).....	54
3.2	Naïve attack: Based on the foreground pixels, I find the longest horizontal distance (white line) and the mean value of vertical area (the respective bounding boxes above).	55
3.3	High-level overview of my attack. (This, and other figures, are best viewed in color.) ...	57
3.4	The circles depict salient features. These salient features are usually corners of an object or texture areas.	58
3.5	(Top): Initial optical flow. (Middle): salient points with short trajectories in background are discarded. (Lower): Trajectories on non-codeword characters are also discarded.	60
3.6	Example foreground extraction.....	61
3.7	re-running tracking with a lower threshold on corner quality: Left: before modification. Right: after modification.	61
3.8	Left: before segmentation. Right: trajectories are marked with different colors and bounding boxes are calculated based on the center of the trajectories and the orientation of the points. The red points denote areas with no trajectories.	62
3.9	Iterative decoding of a CAPTCHA.	63
3.10	Extended case. Top: scrolling; bottom: in-place.....	66
3.11	Overlapping characters (with ratio = 0.49).	67
3.12	Semi-transparent: 80% background to 20% foreground pixel ratio. (Best viewed in color.)	67
3.13	Emerging CAPTCHA. (a) Top: noisy background frame. Middle: derivative of foreground image. Bottom: single frame for an Emerging CAPTCHA. (b) Successive frames.	68
3.14	Attack success as a function of codeword length.	69

3.15	Attack success rate against <i>Overlapping</i> and <i>Semi-Transparent</i> defenses. Sensitivity refers to the overlap ratio (circles) or the background-to-foreground ratio (squares).	71
3.16	Three backgrounds used for the challenges, shown for the <i>Semi-Transparent</i> variant. ...	72
3.17	Mean number of success, error, and skipped outcomes for <i>Standard</i> , <i>Extended</i> , <i>Overlapping</i> , <i>Semi-Transparent</i> and <i>Emerging</i> variants, respectively.	75
3.18	Time taken to solve the MBOR CAPTCHAs.	77
3.19	Location of errors within the codewords.	78
3.20	Likert-scale responses: 1 is most negative, 10 is most positive.	78
4.1	Overview of my proposed approach.	84
4.2	Examples of facial landmark extraction	86
4.3	Illustration of identity axes (heavy-set to thin) and expression axes (pursed lips to open smile).	87
4.4	3D facial model (right) built from facial landmarks extracted from four images (left).	89
4.5	Naïve symmetrical patching (left); Planar Poisson editing (middle); 3D Poisson editing (right).	91
4.6	Animated expressions. From left to right: smiling, laughing, closing the eyes, and raising the eyebrows.	92
4.7	Example setup using virtual reality to mimic 3D structure from motion. The authentication system observes a virtual display of a user’s 3D facial model that rotates and translates and the device moves. To recover the 3D translation of the VR device, an outward-facing camera is used to track a marker in the surrounding environment.	94
4.8	Spoofing success rate with texture taken from photos of different resolution.	100
4.9	Spoofing success rate with different yaw angles. Left: Using only a single image at the specified angle. Right: Supplementing the single image with low-resolution frontal views, which aid in 3D reconstruction.	101

5.1	Some example threat scenarios that we investigated. Top left: Through reflection from sunglasses and toaster; Top right: Through reflection from eyeball and mirror; Bottom left: Through reflection from sunglasses; Bottom Right: Direct long-distance view. Note: As with the remaining figures in this work, this image is best viewed in color.	107
5.2	Overall design depicting the stages of my approach. First I track the motion of the mobile device (Stage ❶). Then the mobile device is aligned to a template (Stage ❷). In the stabilized image the finger is extracted (Stage ❸) and its fingertip trajectory is computed (Stage ❹). From the trajectories the likely pressed keys are identified (Stage ❺). As an optional step, I apply a language model to improve the quality of the reconstructed text (Stage ❻).	109
5.3	The captured image showing a reflection (of the victim’s screen) as observed in the eyeball.	110
5.4	Cropped image (left) and alignment result (right) from Stage ❷ (overlaid with a reference device layout).	113
5.5	Input aligned image (left) and finger region extraction (right). Shown in the right image is the ellipse fitting (blue ellipse), segmentation (cyan), points with high curvature (yellow), detected fingertip (red circle), and an overlaid reference device layout.	115
5.6	Fingertip trajectory (left) and line modeling (right). The red circles indicate stopping positions for the word “hello”, where the size of the circle corresponds to uncertainty in the stopping position.	116
5.7	METEOR scores for my approach, broken down by scenario.	121
5.8	Left: the cellphone image reflects from the user’s sunglasses, off the toaster’s surface, and is then captured by the camera. Right: the cellphone image reflects from user’s eyeball, off the mirror, and then is captured by the camera.	123
5.9	Example double-reflection from sunglasses and a toaster. Left: Captured image; Right: (correctly) identified keypress (T, with confidence 0.55). The brightness of the detected image (right) is adjusted for viewing convenience and has been overlaid with a reference device layout.	124
5.10	Plot of captured RMS contrast versus METEOR score for the three different scenarios (direct, single-reflection, and repeated reflections).	125
5.11	Example double-reflection from an eyeball and mirror. Left: Captured image; Right: candidate keys (I with confidence 0.10, L with confidence 0.07, and the correct key O with confidence 1.00). The right image has been overlaid with a reference device layout.	126

5.12	Example single reflection from 10m. Left: Single reflection from 10m (correctly predicting \mathbb{I} with confidence 1.0). Right: Direct view from 50m (correctly predicting \mathbb{T} with confidence 1.0). Both images have been overlaid with a reference device layout.	127
6.1	The high-level workflow of my approach. Features are extracted from the captured video and then compared with features from reference videos in a database. The reference video with the most similar feature is output as the most probable candidate.	133
6.2	The intensity signal (top) and respective features (middle and bottom). For illustrative purposes, the sequences are manually aligned. Noises occur as peaks or masked out peaks in the feature sequence	135
6.3	Depiction of a sliding window for extracting the peak-descriptor. To suppress noise related peaks, peaks below a predefined threshold are ignored. Effective peaks pairs create a histogram and the cumulation is used as the descriptor of the window.	140
6.4	Lab environment (left) and home environment setting (right)	144
6.5	The median ratio within the dataset of 54,000 references.	145
6.6	Rank of correct video in the best case (blue) and worst case (red).	146
6.7	Ratio to second-best under different illumination conditions.	148
6.8	Boxplot of the second-best ratio w.r.t. different screen sizes.	148
6.9	Rank of correct video among libraries of size 1,000 and 4,000.	150
6.10	Rank of correct video (among 54,000 videos).	150
6.11	TV reflection in the room is captured from a distance of 13.5 meters (left). The worst case results (right) are illustrated for different types of videos: TV shows, music and film from top to bottom. All segments longer than 180s were successfully retrieved.	151
6.12	TV reflection in the room is captured from a distance of 70.9 meters(top). The camera and the window are labelled in red(top right). The required capture length is compared with direct view and 13.5 meter reflection (bottom). It takes longer for successful retrieval with longer distance.	153
6.13	Captured image directly from window (left), through vinyl blinds (middle) and through a curtain (right).	154
6.14	(Left): Cameras facing opposite directions; (Right): Encoded intensity signals.....	156

7.1	Copy-reference pairs in the VCDB dataset	159
7.2	Overview of the proposed approach	159
7.3	Left : sliding window on derivative feature. Left middle: local feature with bounded uncertainty w.r.t. temporal adjustments. Right middle: local feature in uploaded video and (right) its closest neighbor in the reference set.	162
7.4	Final result of comparing the uploaded video and the selected reference video. Green: matched segment; Blue: Unmatch segment; Red: reference sequence.	164
7.5	Left: the Precision-Recall curve of my approach and TNP on the VCDB +50,000 dataset., Right: The Precision-Recall curve for TNP and my approach for different genres of videos.	165
7.6	Left: Precision-Recall curve for my approach with(red)/without(purple) 3×3 tiling, Xu <i>et al.</i> (Xu et al., 2014a) (green) and TNP (blue). Right: Precision-Recall curve of the TNP approach vs my approach with different accuracies.....	167
7.7	The influence of query length on computation cost.....	170

LIST OF ABBREVIATIONS

3DMM	3D morphable model
AFSA	American Forces Security Agency
CAPTCHA	Completely Automated Public Turing Test to Tell Computers and Humans Apart
CBCD	Content-based copy detection
COG	Cognitive-based
CQT	Constant-Q transforms
CR	Character-recognition
DTW	Dynamic time warping
FFT	Fast Fourier transformation
HCI	Human-computer interface
IR	Image-recognition
MIOR	Moving-image object recognition
NBS	National Bureau of Standards
NIST	National Institutes of Standards and Technology
OCR	Optical character recognition
OTT	One-time tapes
PCA	Principal component analysis
PDF	Probability density function
SDM	Supervised descent method
SFM	Structure from motion
SFS	Structure from shading
SNR	Signal-to-noise ratio
UGC	User Generated Content
UHF	Ultra High Frequency
VR	Virtual reality

LIST OF SYMBOLS

A_i^{id}	Identity principal axes
A_i^{exp}	Expression principal axes
I_{cap}	Intensity of the video frame that is captured
I_{ref}	Intensity of the video frame that is displayed on the screen
I_{noise}	Intensity of noise
P_0	Unit area emanation power
S_{screen}	Size of the screen
$s_{i,j}$	Point coordinate on captured image
$S_{i,j}$	Point coordinate on 3D object
P	Projection matrix
R_j	3D rotation matrix
t_j	Translation vector
α_{cap}	Percentage of light captured by the camera
α_{cam}	Sensitivity of camera sensor
α^{id}	Identity weight vector
α^{expo}	Expression weight vector

CHAPTER 1: INTRODUCTION

Video analysis and video surveillance are active areas of computer vision research. Video analysis techniques are widely used in a wide range of tasks, including recognizing license plate numbers, detecting pedestrians, and identifying copyright infringement. Nowadays, staggering amounts of videos were captured and uploaded to the internet. For example, in 2015, over 300 hours of video are uploaded to YouTube every minute (Youtube, 2015), the number of video posts per Facebook user has increased by more than 75 percent in the past year alone, and nearly 70 million photos and videos are posted on Instagram each day¹. Analysis of these video datasets is an important and financially lucrative task for both online video companies and their customers. For instance, in 2015, YouTube reported that more than 5,000 partners use Content ID (a large-scale copy detection application for online videos), and the service has led to over \$1 billion in payouts to creators and rights holders since its launch in 2007.

The staggering amount of video data also poses serious challenges for computer vision techniques and received much attention from academia. Two key tasks for large scale video analysis and surveillance are video-based event detection and large-scale video event retrieval. The goal of video event detection is to localize and identify spatio-temporal patterns in video, such as characters (Bhardwaj and Mahajan, 2015), key strokes (Balzarotti et al., 2008a), moving vehicles (Jazayeri et al., 2011) and pedestrians (Dollar et al., 2012). While event detection approaches extract event information from each video, it still remains a difficult task to examine a large-scale dataset and find a particular a certain video event, which is called “event retrieval”. Given the astronomic amount of online video resources, sophisticated fingerprint extraction and indexing schemes must be designed for efficient video event retrieval.

¹ See <https://instagram.com/press/>. Accessed May 8, 2015.

For video event detection, the traditional approach is to track the targets, stabilize tracking results, and then recognize them (Efros et al., 2003; Ikizler and Forsyth, 2007; İközler and Forsyth, 2008). The detected action is recognized by the subtle changes in the stabilized local frame patches, which are usually noisy and of low resolution, resulting in limited detection accuracy. Other approaches utilize spatial-temporal volumes to recognize action patterns (Ke et al., 2010; Rodriguez et al., 2008; Derpanis et al., 2010; Laptev and Pérez, 2007; Ke et al., 2007; Jiang et al., 2006, 2010; Seo and Milanfar, 2009). These approaches treat the temporal dimension of the video in the same manner as the spatial dimensions and develop volume features in the high-dimensional space. They rely on sliding windows to perform exhaustive searching for events, which is computationally expensive for large-scale datasets.

For video event retrieval, the traditional approach is to utilize spatial features such as SIFT/SURF (Gengembre et al., 2008; Ke et al., 2004; Zhang and Chang, 2004) to match individual frames in the target video and frames from the dataset. However, spatial features are sensitive to spatial-transformations, noise, and resolution changes, making them less usable in certain scenarios such as copy detection (Jiang et al., 2014). Moreover, the approaches using spatial features require matching of every key frame, making these less efficient for large-scale datasets.

While the aforementioned techniques can be applied successfully in surveillance scenarios such as traffic and pedestrian detection, it is still an open problem to utilize these techniques for security analysis under various adversarial constraints. By analysing video events instead of still images or photos, additional temporal information can be extracted, and that information could be leveraged by both the attackers and the defenders. For instance, while it has been known that reflections can be used to reconstruct text on the screen (Kuhn and Kuhn, 2003; Backes et al., 2008), these approaches rely on expensive telescopes with large aperture, making them impractical in the real world. Most recently, Raguram et al. (2011) proposed an approach for analyzing pop-out events from videos instead of decoding text on the screen directly. In this way, off-the-shelf devices can be used to achieve the same precision and attack range as previous results from Backes et al. (2008), significantly reducing the gap between theory and practice.

Moreover, in some adversarial settings the existing techniques fail to deal with practical challenges. For instance, if the adversary only acquires views of the victim through multiple reflections or eyeball reflections, the event detection scheme will have to deal with extremely low resolution, high noise and significant occlusion, which is beyond the scope of traditional event detection methods. In these cases, I extend existing approaches or seek novel ideas that are more robust and efficient. Towards this goal, I also make contributions in the computer vision domain by introducing cues and features that improve the robustness of existing techniques.

To summarize, in my research (Chapter 3 to Chapter 7), I have investigated realistic security settings for both video event detection and retrieval. For video event detection, I robustly detect codewords from moving-object CAPTCHAs (Completely Automated Public Turing Test to Tell Computers and Humans Apart) and reconstruct typed input from repeated reflections. I also propose a novel technique that defeats existing face liveness detection systems. For video event retrieval, from the attacker’s perspective, I utilize temporal cues to infer TV content from light effusions. On the other hand, from the defender’s perspective, I use similar video retrieval techniques to retrieve copies from large video collections. In each of these scenarios, I analyze the advantages and limitations of the adversary and develop robust computer vision techniques to meet various challenges.

1.1 Thesis Statement

The trajectories of objects can be utilized to improve the robustness of visual event detection against low resolution, noise, and occlusion, making it possible to decode overlapping characters in motion (*e.g.*, video CAPTCHA) and even reconstruct typing behavior from reflections in an eyeball. Features based on temporal brightness changes can be used as a resilient feature for event retrieval, enabling TV content retrieval from light effusions and robust copy detection against practical space-temporal transformations. These techniques can be used to solve computer vision challenges in many security scenarios (*e.g.*, video CAPTCHA decoding, compromising emanation and copy detection) with demanding adversarial restrictions.

1.2 List of Innovations

In summary, the primary contributions of this dissertation are as follows:

1.2.1 Decoding Codewords from Motion-based Captchas

In Chapter 2, given a moving-image-object-recognition (MIOR) CAPTCHA, I extract the motion contained in the video using the concept of salient features. To infer the motion of the salient feature points, I apply object tracking techniques. With a set of salient features at hand, I then use these features to estimate the color statistics of the background. Next, to account for the fact that all characters of the codewords move independently, I segment the foreground into n segments as in the naïve attack. I select each image patch containing a candidate character and evaluate the patch using a neural network based classifier (Simard et al., 2003). The classifier outputs a likelihood score that the patch contains a character. As a final enhancement, I incorporate a feedback mechanism in which I use high confidence inferences to improve low-confidence detections of other patches. The net effect is that I reduce the distractions caused by mutually overlapping characters. Once all segments have been classified, I output a hypothesis for all the characters in the codeword.

Contributions in computer vision. My attack inherently leverages temporal information in the moving-image object recognition (MIOR) CAPTCHAs and also exploits the fact that only recognition of known objects is needed. My methods also rely on a reasonably consistent appearance or slowly varying appearance over time. That said, they can be applied to any set of known objects or narrowly defined objects under affine transformations that are known to work well with detection methods in computer vision (Viola and Jones, 2001).

Contributions in security. I show that not only are there inherent weaknesses in the current MIOR CAPTCHA design, but that several obvious countermeasures (e.g., extending the length of the codeword) are not viable attack countermeasures. More importantly, my work highlights the fact that the choice of underlying hard problem by NuCaptcha’s designers was misguided; its particular implementation falls into a solvable subclass of computer vision problems.

Publications. Xu, Y., Reynaga, G., Chiasson, S., Frahm, J.-M., Monroe, F., and van Oorschot, P. C. (2012). Security and usability challenges of moving-object captchas: Decoding codewords in motion. In *USENIX Security Symposium, Bellevue, WA*, pages 49–64,

Xu, Y., Reynaga, G., Chiasson, S., Frahm, J.-M., Monroe, F., and van Oorschot, P. C. (2014b). Security analysis and related usability of motion-based captchas: Decoding codewords in motion. *Dependable and Secure Computing, IEEE Transactions on*, 11(5):480–493.

1.2.2 Defeating Face Liveness Detection with Virtual Models

In Chapter 4, I introduce a novel approach to bypass modern face authentication systems. More specifically, by leveraging a handful of pictures of the target user taken from social media, I show how to create realistic, textured, 3D facial models that undermine the security of widely used face authentication solutions. My framework makes use of virtual reality (VR) systems, incorporating along the way the ability to perform animations (*e.g.*, raising an eyebrow or smiling) of the facial model, in order to trick liveness detectors into believing that the 3D model is a real human face. The synthetic face of the user is displayed on the screen of the VR device, and as the device rotates and translates in the real world, the 3D face moves accordingly. To an observing face authentication system, the depth and motion cues of the display match what would be expected for a human face. This work was submitted to the 25th USENIX Security Symposium, 2016.

Contributions in computer vision. I show that it is possible to undermine modern face authentication systems using one such attack. Moreover, I show that an accurate facial model can be built using *only* a handful of publicly accessible photos — collected, for example, from social network websites — of the victim. From a pragmatic point of view, I am confronted with two main challenges: *i*) the number of photos of the target may be limited, and *ii*) for each available photo, the illumination setting is unknown and the user’s pose and expression are not constrained. To overcome these challenges, I leverage robust, publicly available 3D face reconstruction methods from the field of computer vision, and adapt these techniques to fit my needs.

Contributions in security. I argue that such VR-based spoofing attacks constitute a fundamentally new class of attacks that point to a serious weaknesses in camera-based authentication systems: unless they incorporate other sources of verifiable data, systems relying on color image data and camera motion are prone to attacks via virtual realism. To demonstrate the practical nature of this threat, I conduct thorough experiments using an end-to-end implementation of my approach and show how it undermines the security of several face authentication solutions, including both motion-based and liveness detectors.

1.2.3 Reconstructing Typed Input from Repeated Reflections

In Chapter 5, I propose a method that successfully transcribes the text typed on a keyboard by exploiting video of the user typing, either directly or through up to two reflections from nearby objects. To do so, I devise techniques that are resistant to low resolution, blurring, and noise. I take as input a recording of the target device while the user types on its keyboard. First, I roughly estimate the location of the device in the image. Next, the image of the device is aligned to a known reference template of the device’s keyboard layout. Then, the fingertips are identified in the video and the locations of the fingertips over the video frames are combined into trajectories. These trajectories are then analyzed to identify the pressed keys. From these pressed keys I then reconstruct the typed text. Finally, as an optional post-processing step, I apply a language model in order to improve the readability of the final text. Aside from some initial input in the tracking stage to identify the first frame containing the target device, the entire process is fully automated.

Contributions in computer vision. My approach works despite low resolution images (a natural consequence of increasing the distance between target and camera) and high noise levels (due to reduced light levels in reflections of objects as compared to the originals). To overcome these challenges and achieve my goals, I extend a large body of work on object detection and tracking in the area of computer vision. Specifically, I extend existing finger tracking mechanisms to consider spatial context in order to reliably locate fingertips in the images.

In addition, I propose a novel method for identifying fingertip trajectories based on robust estimation.

Contributions in security. Tracking fingers, rather than recognizing displayed images, broadens the class of vulnerable devices to include Apple’s iPad and similar ones, which do not use any pop-out effect. Moreover, my approach is capable of operating at significant distances from the victim (e.g., up to 50 meters away with a camcorder). Perhaps most importantly, my approach operates effectively even for *repeated reflections*, i.e., *reflections of reflections* in nearby objects. This feat allows us to reconstruct typed input from the image of a mobile phone’s screen on a user’s eyeball as reflected through a nearby mirror, extending the privacy threat to include situations where the adversary is located *around a corner* from the user.

Publication Xu, Y., Heinly, J., White, A. M., Monroe, F., and Frahm, J.-M. (2013). Seeing double: reconstructing obscured typed input from repeated compromising reflections. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security, Berlin, Germany*, pages 1063–1074. ACM

1.2.4 Inferring TV Content from Light Effusions

In Chapter 6, I leverage the key insight that the observable emanations of a display (e.g., a TV or monitor) during presentation of the viewing content induces a distinctive flicker pattern that can be exploited by an adversary. I compute the average pixel brightness of each frame in the video, resulting in a mean brightness signal for the video. To capture the sharp changes in brightness, I then use the gradient of the signal as the descriptor for the video. Similar to the captured video, every video in the adversary’s collection is represented by a feature based on the gradient of the brightness signal. Note that while the mean brightness signal of the reference video and the captured videos signal may vary, their gradient-based features share common characteristics, and it is those commonalities that are used to identify the content being watched.

Contributions in computer vision. The key contribution in this work lies in the techniques I use to take advantage of temporal information to find matches among reference and capture signals, even in the face of significant noise and signal distortions. To do this, I extend traditional correlation measures to utilize temporal information when computing similarity scores between sequences. The resulting strategy significantly outperforms traditional correlation measures (e.g., Greveler et al. (2012)).

Contributions in security. My proposed approach works successfully in a number of practical scenarios, including (but not limited to) observations of light effusions through the windows, on the back wall, or off the victim’s face. My empirical results show that I can successfully confirm hypotheses while capturing short recordings (typically less than 4 minutes long) of the changes in brightness from the victim’s display from a distance of 70 meters.

Publication Xu, Y., Frahm, J.-M., and Monroe, F. (2014a). Watching the watchers: Automatically inferring tv content from outdoor light effusions. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, Scottsdale, AZ*, pages 418–428. ACM

1.2.5 Detecting Copies from Large Video Collections

In Chapter 7, I propose a novel content-based copy detection (CBCD) approach that consists of two main components: a knowledge extraction stage and a video subsequence matching (or “copy detection”) stage. In the knowledge extraction stage, I first preprocess the reference library by extracting the derivative of the average intensity for all its videos (i.e., computing their “fingerprints”). These derivative features encode the temporal position of illumination changes in the videos, which can be leveraged to find linkages among a collection of videos. Then, my proposed approach focuses on significant illumination changes that result in distinct derivative features. These local temporal features (that cover a short span of time around a peak) are extracted for each reference video and encode the gradient signal profile in close proximity to significant illumination change in a manner

that is robust to temporal transformations. In that way, the local features of a transformed video will be equivalent to those in the corresponding original video. To make finding matches more efficient, a K-d tree index structure is built upon the local features of all reference videos. Then, the detection stage is used to determine if the video under examination contains infringing material. In this phase, the derivative features and the local temporal features of the uploaded video are computed. The local features are then used as a first approximation. The collection of possible matches is then considered for further scrutiny using full video information. This work was submitted to ECCV'16.

Contributions in computer vision. I propose a novel copy-detection system utilizing temporal information. Our large-scale empirical results show that my approach achieves higher accuracy and recall rate on real-world videos from the VCDB dataset than contemporary approaches. I am able to achieve improved accuracy because I am able to leverage observations regarding temporal consistency for practical content-based copy detection, which is difficult to thwart without significantly degrading the user's viewing experience. My approach is more robust against spatial transformations and commonly used temporal transformations.

Contributions in security. My technique narrows the detection gap in the important area of temporal transformations applied by would-be pirates. My large-scale evaluation on real-world data shows that I can successfully detect infringing content from movies and sports clips with 90.0% precision at a 71.1% recall rate, and can achieve that accuracy at an average time expense of merely 5.3 seconds, outperforming the state of the art by an order of magnitude in terms of speed.

CHAPTER 2: BACKGROUND AND RELATED WORK

My work spans four different security scenarios: the security of moving object CAPTCHAs (Completely Automated Public Turing Test to Tell Computers and Humans Apart), the security of face liveness detection, compromising optical emanation, and content-based video copy detection. In each of these scenarios, both the adversary and the defender have different constraints and challenges. To better understand the problems, in the following sections, I first review the related work in video event detection and retrieval. Then, I provide a review of background and related work in each adversary constraint.

2.1 Video Event Detection and Retrieval

Before diving into specific scenarios, I first review general developments in video event detection and retrieval.

2.1.1 Video Event Detection

Video event detection has attracted significant research interest in recent years. This trend dates back to the early 1990s when researchers first tried to analyze movements in videos (Aggarwal et al., 1994; Cedras and Shah, 1994, 1995), mainly using optical flow methods (Horn and Schunck, 1981). In the late 1990s, the most widely discussed topic in video event detection was the recognition of hand gestures (Pavlovic et al., 1997), for which most used dynamic time warping or Hidden Markov Models (Eddy, 1996) for event detection. Later in the early 2000s, given the rapid developments in recognition and classification techniques, a basic framework was setup for video event detection in general (Wang et al., 2003), which consisted of motion segmentation, object classification, object tracking, action recognition, and semantic description. Classification techniques such as neural

networks and Principal Component Analysis were also introduced for behavior understanding. Since different scenarios may have dramatically different constraints and requirements for video event detection, in subsequent research efforts this framework was further developed to adapt to different scenarios.

For behavior analysis in video surveillance for security, the framework can be modified by adding a person identification component (Ko, 2008). After detecting a human target, the person's image can be automatically fed into an indexing scheme for identification. Then, the movement trajectory of the target is analyzed for abnormality. This application is particularly useful for security and abnormal behavior detection.

For pedestrian behavior analysis, Geronimo et al. (2009) discussed recent developments in pedestrian detection systems. The main challenge in this scenario is the variety of human appearance (*e.g.*, different clothes, changing size, aspect ratio, and dynamic shape) and the unstructured environment. The basic structure in this scenario still follows the one proposed by Wang et al. (2003), which consists of background segmentation, classification, and tracking. Due to the requirement of robustness, refinement techniques are often used to improve the result from the object classification phase. For instance, by analyzing the movement patterns of a pedestrian, the system can automatically identify abnormal behavior, and further analyze if it is a mis-classification or not.

Lastly, human-computer interfaces (HCI) are also a widely discussed topic in video event detection. Pantic et al. (2007) reviewed the latest developments in this field. Techniques such as expression recognition, face recognition, and gesture detection all have significant applications in HCI. Fortunately, in these cases, the human is more or less cooperating with the computer, making it possible to assume a relatively predictable environment, *e.g.*, with sufficient resolution and brightness level.

In my work, I also mostly follow the framework of Wang et al. (2003) and develop different video event detection schemes that specifically adapt to different scenarios, including decoding codewords from motion-based CAPTCHAs (Section 2.2) and reconstructing typed input from

repeated reflections (Section 2.4.1.1). On the other hand, I also explore the potential security weakness of existing face authentication systems by proposing a novel adversary model (Section 2.3).

2.1.2 Video Event Retrieval

Video event retrieval first originated in the 1990s when techniques were proposed that index video events by detecting visual objects and assigning icons for each shot (Aslandogan and Yu, 1999). However, these approaches rely on the results of video event detection and are limited to targets that can be robustly detected and tracked.

In 2007, Jiang et al. (2007) proposed to use bag-of-features for semantic video retrieval. With their approach, videos of similar topics can be retrieved, not limited to certain objects. However, Jiang et al. (2007) failed to take temporal information into consideration, which means their approach does not distinguish between different temporal events. For instance, the proposed system cannot tell the difference between an object moving left and an object moving right. This fact restrains Jiang et al. (2007)’s approach to temporally irrelevant events only.

Later, Jones and Shao (2013) extended the bag-of-features approach (Jiang et al., 2007). Temporal information was taken into consideration using their proposed technique called spatial-temporal pyramid, which includes temporal information in the indexing structure. Their approach achieved over 95% retrieval accuracy on the UCF Sports dataset (Soomro and Zamir, 2014). However, the accuracy is lower than 30% on the HOHA2 dataset (Marszałek et al., 2009), partly because of the frequent scene switches.

Audio and subtitle information are also sometimes used for video event retrieval. For instance, Yang and Meinel (2014) propose to use audio and text information for speech retrieval. However, sometimes the user is only interested in visual content without speech, making this category of approaches constrained to certain scenarios.

Lastly, copy detection is also a hot topic in video event retrieval. Interestingly, while much work claiming near perfect results on synthetic datasets, I show that these techniques do not work well on real-world data. I return to this later in Section 2.5.

In my work, I use temporal features only for video retrieval. In this way, my proposed system is immune to distortions, noise, and most spatial transformations, and maintaining a superior processing speed. I first propose this technique to automatically infer TV content from light effusions (Section 2.4.1.2). Later I extend this technique to content-based video copy detection (Section 2.5). In the remainder of this chapter, I discuss the specific background for each of the four scenarios: (i) the security of moving object CAPTCHAs (Section 2.2); (ii) the security of face liveness detection (Section 2.3); (iii) compromising optical emanation (Section 2.4); (iv) content-based video copy detection (Section 2.5).

2.2 The Security of Moving Object CAPTCHAs

CAPTCHAs (Completely Automated Public Turing Test to Tell Computers and Humans Apart) first came into place in 2003, when Von Ahn et al. (2003) used distorted letters to distinguish between computers and humans. Once invented, it quickly became popular for websites to avoid attacks such as automated spamming and brute force password guessing.

To date, a myriad of text, audio, and video-based CAPTCHAs have been suggested (Hidalgo and Alvarez, 2011), many of which have succumbed to different attacks (Golle, 2008; Mori and Malik, 2003; Yan and Ahmad, 2007, 2008a; Zhu et al., 2010; Bursztein et al., 2011b,a). While text-based CAPTCHAs that prompt users to recognize distorted characters have been the most popular form to date, motion-based or video CAPTCHAs that provide some form of moving challenge have recently been proposed as the successor to static CAPTCHAs. One prominent and contemporary example of this new breed of CAPTCHAs is NuCaptcha (NuCaptcha, 2011), which asserts to be “*the most secure and usable CAPTCHA*,” and serves millions of video CAPTCHAs per day. The general idea embodied in these approaches is to exploit the remarkable perceptual abilities of humans, which allows us to quickly unravel complex patterns in dynamic scenes. For example, in the

case of NuCaptcha, users are shown a video with a series of characters (so-called random *codewords*) moving across a dynamic scene, and solve the CAPTCHA by identifying the characters of the codeword. For enhanced security, the codewords are presented among adversarial clutter (Mori and Malik, 2003) (e.g., moving backgrounds and other objects with different trajectories), and consecutive characters may even overlap significantly.

In this section, I first review the adversary model in this scenario and the underlying assumption for moving object CAPTCHA to work. Then, I review related work that covers the standard approaches of CAPTCHA decoding. Finally, I analyse the remaining challenges for the security of moving object CAPTCHAs.

2.2.1 Adversary Modeling

The potential adversary in this scenario acquires a video CAPTCHA, which consists of a sequence of video frames, and intends to decode the CAPTCHA and extract the codewords automatically. For moving object CAPTCHA to be secure, the underlying assumption is that attacks based on state-of-the-art computer vision techniques are likely to fail at uncovering these challenges within video sequences, whereas real users will be able to solve the challenges with little effort.

To further understand this assumption, I review some basic concepts in human visual cognition. In the human brain, it is generally assumed that an image is represented by the activity of “units” tuned to local features (e.g., small line and edge fragments). It is also widely believed that objects appearing in a consistent or familiar background are detected more accurately, and processed more quickly, than objects appearing in an inconsistent scene (Oliva and Torralba, 2007). In either case, a person must somehow separate as much as possible of the image once they see it. This feat is believed to be done via a *segmentation* process that attempts to find different objects in the image that “go together” (Ullman, 2000).

As with other aspects of our visual system, segmentation involves different processes using a multitude of sources of information (e.g., texture and color), which makes it difficult to establish which spatial properties and relations are important for different visual tasks. While there is

evidence that human vision contains processes that perform grouping and segmentation prior to, and independent of, subsequent recognition processes, the exact processes involved are still being debated (Oliva and Torralba, 2007).

Given the complexity of the visual system, it is widely believed that this feat remains unmatched by computer vision algorithms. One of the many reasons why this task remains elusive is that perception of seemingly simple spatial relations often requires complex computations that are difficult to unravel. This is due, in part, to the fact that object classification (that is, the ability to accurately discriminate each object of an object class from all other possible objects in the scene) is computationally difficult because even a single individual object can already produce an infinite set of different images (on the retina) due to variations in position, scale, pose, illumination, etc. Discriminating objects of a certain class is further complicated by the often very large inner class variability, which significantly changes the appearance beyond the factors encountered for a single object. Hence, vision operates in a high-dimensional space, making it difficult to build useful forms of visual representation.

In computer vision, the somewhat simpler process of recognizing known objects is simulated by first analyzing an image locally to produce an edge map composed of a large collection of local edge elements, from which we proceed to identify larger structures. In my work, I am primarily interested in techniques for object segmentation and tracking. In its simplest form, *object tracking* can be defined as the problem of estimating the trajectory of an object in the image plane as it moves around a scene. Tracking makes use of temporal information computed from a sequence of frames. This task can be difficult for computer vision algorithms because of issues related to noise in the image, complex object motion, the nonrigid nature of objects, etc. However, the tracking problem can be simplified if one can assume that object motion is smooth, the motion is of constant velocity, or that the number and the size of the objects or even appearance and shape information are known. In NuCaptcha, for example, many of these simplifications hold, and so several features (e.g., edges, optical flow) can be used to help track objects. The correspondence search from one frame to the next is performed by using tracking.

In video, this correspondence can be achieved by building a representation of the scene (called the *background model*) and then finding deviations from the model for each incoming frame. Intuitively, any significant change in the image region from the background model signifies a moving object. The pixels constituting the regions undergoing change are marked for further processing, and a connected component algorithm is applied to obtain connected regions. This process is typically referred to as *background subtraction*. At this point, all that is needed is a way to partition the image into perceptually similar regions, and then infer what each of those regions represent.

2.2.2 Related Work

Most CAPTCHAs in commercial use today are character-recognition (CR) CAPTCHAs involving still images of distorted characters; attacks essentially involve building on optical character recognition advances. Audio CAPTCHAs (AUD) are a distinct second category, though unrelated to my present work. A third major category, image-recognition (IR) CAPTCHAs, involves classification or recognition, of images or objects other than characters. A well-known example, proposed and then broken, is the Asirra CAPTCHA (Elson et al., 2007; Golle, 2008) which involves object classification (e.g., distinguishing cats from other animals such as dogs). CR and IR schemes may involve still images (CR-still, IR-still), or various types of dynamic images (CR-dynamic, IR-dynamic). Dynamic text and objects are of main interest of my work, and contribute to a cross-class category: moving-image object recognition (MIOR) CAPTCHAs, involving objects in motion through animations, emergent-image schemes, and video (NuCaptcha, 2011; Liao and Chang, 2004; Cui et al., 2009; Shirali-Shahreza and Shirali-Shahreza, 2008; Cui et al., 2010a,b; Mitra et al., 2009). A fourth category, cognitive-based CAPTCHAs (COG), include puzzles, questions, and other challenges related to the semantics of images or language constructs. I include here content-based video-labeling of YouTube videos (Kluever and Zanibbi, 2009).

The most comprehensive surveys of CAPTCHAs to date are those by Hidalgo and Maranon (Hidalgo and Alvarez, 2011) and Basso and Bergadano (Basso and Bergadano, 2010). I

also note other summaries: for defeating classes of AUD CAPTCHAs, Soupionis (Soupionis and Gritzalis, 2010) and Bursztein et al. (Bursztein and Bethard, 2009; Bursztein et al., 2011a); for defeating CR CAPTCHAs, Yan et al. (Yan and Ahmad, 2007; Yan and El Ahmad, 2011) and Bursztein (Bursztein et al., 2011b); for a systematic treatment of IR CAPTCHAs and attacks, Zhu et al. (Zhu et al., 2010), as well as for robustness guidelines.

Usability has also been a central focus, for example, including a large user study of CR and AUD CAPTCHAs involving Amazon Mechanical Turk users (Bursztein et al., 2010), a user study of video-tagging (Kluever and Zanibbi, 2009), usability guidelines and frameworks related to CR CAPTCHAs (Yan and Ahmad, 2008b). Chellapilla et al. (Chellapilla et al., 2005b,a) also address robustness. Hidalgo et al. (Hidalgo and Alvarez, 2011) and Bursztein et al. (Bursztein et al., 2011b) also review evaluation guidelines including usability. Research on underground markets for solving CAPTCHAs (Motoyama et al., 2010), and malware-based CAPTCHA farms (Egele et al., 2010), raise interesting questions about the long-term viability of CAPTCHAs.

Concurrent to my own work, Bursztein (Bursztein, 2012) presented an approach to break the video CAPTCHAs used by NuCaptcha. The technique exploits the video by treating it as a series of independent frames, and then applies a frame-based background removal process (Bursztein et al., 2011b) to discard the video background. Next, frame characteristics (e.g., spatial salient feature density and text aspect ratio of the overlapping letters) are used to detect the codeword, after which a clustering technique is used to help segment the characters of the codeword. As a final step, traditional CR-still based attacks are used to recognize the characters in each of the segmented frames. Some stages of the approach taken by Bursztein have similarities to my baseline method which uses single frame segmentation and recognition. In contrast, my subsequent techniques inherently use temporal information contained in the video to identify the codeword, to improve the segmentation, and to enhance the recognition accuracy during the codeword recovery process.

2.2.3 Remaining Challenges

In general, CAPTCHAs are challenges that are difficult for computer and easy for human. However, with the rapid developments in computer algorithms, the ability of a computer is surpassing or getting close to a human's level almost in every domain. This trend is making challenges that used to be hard for computers become feasible, which generates potential vulnerabilities in existing CAPTCHAs. Therefore, it is always an important task to re-evaluate the security of contemporary CAPTCHAs, using the latest developments in computer algorithms.

For moving object CAPTCHAs particularly, although it is still widely believed that the cognitive perception of human is beyond the ability of current computer algorithms, it remains unclear whether the current state of moving object CAPTCHAs utilizes this gap successfully. Namely, are the current moving object CAPTCHAs challenges hard for computers while easy for humans? Additionally, with possible variations of moving object CAPTCHA (*e.g.*, extending codeword length, increasing overlapping), will they better satisfy the goal of telling humans and computers apart? The answer to these questions leads to a comprehensive study in both the security and the usability of moving object CAPTCHAs, which is a contribution of my work.

2.3 The Security of Face Liveness Detection

While fingerprints has long been utilized as an authentication method since as early as 500 B.C. (Jain and Kumar, 2010), modern automatic biometric systems only emerge in the 1890s when the automated fingerprint systems were first used for criminal identification. Legal standards were eventually established in 1986 by The National Bureau of Standards (NBS), now the National Institutes of Standards and Technology (NIST) (Karu and Jain, 1996).

The history of automated face authentication is even shorter. In 1991, after the eigenface technique (Turk and Pentland, 1991) was first developed, different face recognition techniques such as feature-based recognition (Samal and Iyengar, 1992), elastic matching (Chellappa et al., 1995) and neural nets (Valentin et al., 1994) significantly broadened the horizon. Since then, face

recognition has been a hot topic in computer vision. In 2014, Lu and Tang (2014)’s work already surpassed human performance in precision and recall. Most recently, recent advances with deep learning algorithms (Taigman et al., 2014; Parkhi et al., 2015) have pushed the boundary even further. However, how to use well-developed face recognition techniques in face authentication is still an open problem, with no formal standard or well-acknowledged approaches. Lenovo, Asus, and Toshiba have all made attempts to use face authentication for authenticating users on desktop computers. However, at the Black Hat security conference in 2009, Duc and Minh (2009) demonstrated that all these systems can be bypassed with photographs and fake pictures of faces. Similar weaknesses have been shown with Android OS, which augmented its face authentication approach in 2012 to require users to blink while authenticating (*i.e.* as a countermeasure to still-image spoofing attacks). Unfortunately, this approach was also shown to provide little protection, and can be easily bypassed by presenting the system with two alternating images — one with the user’s eyes open, and one with her eyes closed.

Despite these failure attempts, face authentication is still a hot research topic. Researchers have realized that the main barrier of face authentication approach is to prevent ”spoofing attacks” as shown earlier. As a result, the latest designs of face authentication systems all contain a *face liveness detection component* that distinguishes real faces from spoofed ones. In the rest of this section, I review the current developments in face liveness detection, and their adversary model and assumptions.

2.3.1 Adversary Modeling

To better understand the design of face liveness detection schemes, I first review the basic model of a potential adversary. Loosely speaking, three types of such spoofing attacks have been used in the past, to varying degrees of success: (*i*) still-image-based spoofing, (*ii*) video-based spoofing, and (*iii*) 3D-mask-based spoofing. As the name suggests, still-image-based spoofing attacks present one or more still images of the user to the authentication camera; each image is either printed on paper or shown with a digitized display. Video-based spoofing, on the other hand, presents a pre-recorded

video of the victim’s moving face in an attempt to trick the system into falsely recognizing motion as an indication of liveness. The 3D-mask-based approach, wherein 3D-printed facial masks are used, was recently explored by Erdogmus and Marcel (2014).

2.3.2 Related Work

Given the three prominent classes of spoofing attacks mentioned earlier, it should be clear that while still-image-based attacks are the easiest to perform, they can be easily countered by detecting the 3D structure of the face. Video-based spoofing is more difficult to accomplish because facial videos of the target user may be harder to come by; moreover, such attacks can also be successfully defeated, for example, using the recently suggested techniques of Li et al. (2015) (which I discuss in more detail later). 3D-mask-based approaches, on the other hand, are harder to counter. That said, building a 3D mask is arguably more time-consuming and also requires specialized equipment. Nevertheless, because of the threat this attack vector poses, much research has gone into detecting the textures of 3D masks (Erdogmus and Marcel, 2014).

Just as new types of spoofing attacks have been introduced to fool face authentication systems, more advanced methods for countering these attacks have been developed. Nowadays, the most popular liveness detection techniques can be categorized as either texture-based approaches, motion-based approaches, or liveness assessment approaches. I discuss each in turn.

Texture-based approaches (Kim et al., 2012a; Määttä et al., 2011; Yang et al., 2013; Tan et al., 2010; Peixoto et al., 2011; Erdogmus and Marcel, 2014) attempt to identify spoofing attacks based on the assumption that a spoofed face will have a distinctly different texture from a real face. Specifically, they assume that due to properties of its generation, a spoofed face (irrespective of whether it is printed on paper, shown on a display, or made as a 3D mask) will be different from a real face in terms of shape, detail, micro-textures, resolution, blurring, gamma correction, and shading. That is, these techniques rely on perceived limitations of image displays and printing techniques. However, with the advent of high-resolution displays (*e.g.*, 5K), the difference in visual quality between a spoofed image and a living face is hard to notice. Another limitation is that these

techniques often require training on every possible spoofing material, which is not practical for real systems.

Motion-based approaches (Kim et al., 2013; Bao et al., 2009; Kollreider et al., 2005; Lagorio et al., 2013; Wang et al., 2013) detect spoofing attacks by using the motion of the user’s head to infer 3D shape. Techniques such as optical flow and focal-length analysis are typically used. The basic assumption is that structures recovered from genuine faces usually contain sufficient 3D information, whereas structures from fake faces (photos) are usually planar in depth. For instance, the approach of Li et al. (2015) checks the consistency of movement between the mobile device’s internal motion sensors and the observed change in head pose computed from the recorded video taken while the claimant attempts to authenticate herself to the device. Such 3D reasoning provides a formidable defense against both still-image and video-based attacks.

Lastly, liveness assessment techniques (Jee et al., 2006; Sun et al., 2007; Kollreider et al., 2007, 2008) require the user to perform certain tasks during the authentication stage. For the systems I evaluated, the user is typically asked to follow certain guidelines during registration, and to perform a random series of actions (*e.g.*, eye movement, lip movement, and blinking) at login. The requested gestures help to defeat contemporary spoofing attacks.

Take-away: For real-world systems, liveness detection schemes are often combined with motion-based approaches to provide better security protection than either can provide on its own. With these ensemble techniques, traditional spoofing attacks can be reliably detected. For that reason, the combination of motion-based systems and liveness detectors has gained traction and is now widely adopted in many commercial systems, including popular face authentication systems offered by companies like KeyLemon, Rohos, and Biometrics.

2.3.3 Remaining Challenges

The current state of the art face liveness detection systems successfully address all the three spoofing models. However, it is unclear whether there exist other spoofing categories that have

yet to be discovered, which would require face authentication systems to be redesigned to be safe against the attack.

In my work, I show the reality of such hazards by proposing a novel spoofing approach that undermines all the existing face liveness detection techniques. More specifically, by leveraging a handful of pictures of the target user taken from social media, I show how to create realistic, textured, 3D facial models that undermine the security of widely used face authentication solutions. My framework makes use of virtual reality (VR) systems, incorporating along the way the ability to perform animations (e.g. raising an eyebrow or smiling) of the facial model, in order to trick liveness detectors into believing that the 3D model is a real human face. The synthetic face of the user is displayed on the screen of the VR device, and as the device rotates and translates in the real world, the 3D face moves accordingly. To an observing face authentication system, the depth and motion cues of the display match what would be expected for a human face.

In the rest of this section, I list additional related works in the security of users' online photos and 3D reconstruction of human faces.

2.3.3.1 Online Photos and Face Authentication

It should come as no surprise that personal photos from online social networks can compromise privacy. Major social network sites advise users to set privacy settings for the images they upload, but the vast majority of these photos are often accessible to the public or set to 'friend-only viewing' (Liu et al., 2011; Kim et al., 2012b; Golder, 2008). Users also do not have direct control over the accessibility of photos of themselves posted by other users, although they can remove ('un-tag') the association of such photos with their Facebook account.

A notable use of social network photos for online security is Facebook's social authentication (SA) system (Hicks, 2011), an extension of CAPTCHAs that seeks to bolster identity verification by requiring the user to identify photos of their friends. While this method does require more specific knowledge than general CAPTCHAs, Polakis et al. (2012) demonstrated that facial recognition

could be applied to a user’s public photos to discover their social relationships and solve 22% of SA tests automatically.

Given that one’s online photo presence is not entirely controlled by the user alone — but by their collective social circles — many avenues exist for an attacker to uncover the facial appearance of a user, even when the user makes private their own personal photos. In an effort to curb such easy access, work by Ilia et al. (2015) has explored the automatic privatization of user data across a social network. This method uses face detection and photo tags to selectively blur the face of a user when the viewing party does not have permission to see the photo. In the future, such an approach may help decrease the public accessibility of users’ personal photos, but it is unlikely that an individual’s appearance can ever be completely obfuscated from attackers across all social media sites and image stores on the Internet.

Clearly, the availability of online user photos is a boon for an adversary tasked with the challenge of undermining face authentication systems. The most germane on this front is the work of Li et al. (2014). There, the authors proposed an attack that defeated commonly used face authentication systems by using photos of the target user gathered from online social networks. Li et al. (2014) reported that 77% of the users in their test set were vulnerable to their proposed attack. However, their work is targeted at face recognition systems that *do not incorporate face liveness detection*. In modern face authentication software, sophisticated liveness detection approaches are already in use, and these techniques thwart still-image spoofing attacks of the kind performed by Li et al. (2014).

2.3.3.2 Defeating Facial Liveness Detection by Building Virtual Models From Public Photos

Constructing a 3D facial model from a small number of personal photos involves the application of powerful techniques from the field of computer vision. Fortunately, there exists a variety of reconstruction approaches that make this task less daunting than it may seem on first blush, and many techniques have been introduced for facial reconstruction from single images (Kemelmacher-Shlizerman and Basri, 2011; Kemelmacher-Shlizerman, 2013; Baumberger et al., 2014; Qu et al., 2014), videos (Suwajanakorn et al., 2014; Shi et al., 2014; Jeni et al., 2015), and combinations of

both (Suwajanakorn et al., 2015). For pedagogical reasons, I briefly review concepts that help the reader better understand my approach.

The most popular facial model reconstruction approaches can be categorized into three classes: shape from shading (SFS), structure from motion (SFM) combined with dense stereoscopic depth estimation, and statistical facial models. The SFS approach (Kemelmacher-Shlizerman and Basri, 2011) uses a model of scene illumination and reflectance to recover face structure. Using this technique, a 3D facial model can be reconstructed from only a single input photo. SFS relies on the assumption that the brightness level and gradient of the face image reveals the 3D structure of the face. However, the constraints of the illumination model used in SFS require a relatively simple illumination setting and, therefore, cannot typically be applied to real-world photo samples, where the configuration of the light sources is unknown and often complicated.

As an alternative, the structure from motion approach (Fidaleo and Medioni, 2007) makes use of multiple photos to triangulate spatial positions of 3D points. It then leverages stereoscopic techniques across the different viewpoints to recover the complete 3D surface of the face. With this method, the reconstruction of a dense and accurate model often requires many consistent views of the surface from different angles; moreover, non-rigid variations (*e.g.*, facial expressions) in the images can easily cause SFM methods to fail. In my scenario, these requirements make such an approach less usable: for many individuals, only a limited number of images might be publicly available online, and the dynamic nature of the face makes it difficult to find multiple images having a consistent appearance (*i.e.* the exact same facial expression).

Unlike SFS and SFM, statistical facial models (Baumberger et al., 2014; Qu et al., 2014) seek to perform facial reconstruction on an image using a training set of existing facial models. The basis for this type of facial reconstruction is the 3D morphable model (3DMM) of Blanz and Vetter (1999, 2003), which learns the principal variations of face shape and appearance that occur within a population, and then fits these properties to images of a specific face. Training the morphable models can be performed either on a controlled set of images (Paysan et al., 2009; Cao et al., 2014) or from internet photo-collections (Kemelmacher-Shlizerman, 2013). The underlying variations fall

on a continuum and capture both expression (*e.g.*, a frowning-to-smiling spectrum) and identity (*e.g.*, a skinny-to-heavy or a male-to-female spectrum). In 3DMM and its derivatives, both 3D shape and texture information are cast into a high-dimensional linear space, which can be analyzed with principal component analysis (PCA) (Jolliffe, 2002). By optimizing over the weights of different eigenvectors in PCA, any particular human face model can be approximated. Statistical facial models have shown to be very robust and only require a few photos for high-precision reconstruction. For instance, the approach of Baumberger et al. (2014) achieves good reconstruction quality using only two images.

To make the process fully automatic, recent 3D facial reconstruction approaches have relied on a few facial landmark points instead of operating on the whole model. These landmarks can be accurately detected using the supervised descent method (SDM) (Xiong and De la Torre, 2013) or deep convolutional networks (Sun et al., 2013). By first identifying these 2D features in an image and then mapping them to points in 3D space, the entire 3D facial surface can be efficiently reconstructed with high accuracy. In this process, the main challenge is the localization of facial landmarks within the images, especially contour landmarks (along the cheekbones) which are half-occluded in non-frontal views; I introduce a new method for solving this problem when multiple input images are available.

2.4 Compromising Optical Emanation

Techniques for undermining a user’s privacy via several forms of compromising emanations have a rich and storied history (*e.g.*, (Highland, 1986; van Eck, 1985; Kuhn and Kuhn, 2003; Asonov and Agrawal, 2004; Zhuang et al., 2005; Elibol et al., 2012; Maggi et al., 2011; Cai and Chen, 2011, 2012; Owusu et al., 2012; Vuagnoux and Pasini, 2009; Backes et al., 2008, 2009)). Yet, the earliest of these ideas originates during the Second World War, in 1943, when Bell engineers accidentally discovered that electrical signals from one-time tapes (OTT’s) were picked up by a nearby oscilloscope. However, this phenomenon did not receive much attention when it was first reported to the Signal Corps. Seven years later the problem was long forgotten when it

was rediscovered in 1951 by the CIA during tests with the very same mixer. They were able to intercept plain text transmissions half a mile along a communications line. The American Forces Security Agency (AFSA) further investigated the problem and discovered an even more serious problem: crypto equipment, connected to radio transmitters, emitted very weak plain text signals. These signals were picked up by the nearby transmitter and modulated onto the transmitted radio waves. The plain text piggy-backed on the transmitters powerful radio waves, making it possible to filter out and read the plain text at distances of tens or hundreds of miles. This was by far the most serious security issue. In early 1960's, after numerous unsuccessful experiments, the Navy Research Laboratory proposed to reduce the voltage and signal amplitude as the final cure of the problem, which led to TEMPEST (not an acronym but sometimes written out as Telecommunications Electronics Material Protected from Emanating Spurious Transmissions). TEMPEST refers to a whole set of technical recourses, standards and regulations to protect communications equipment against unwanted radiation of signals that might be intercepted, analyzed, and exploited by the adversary.

While TEMPEST regulations effectively protect a user's privacy from compromising electrical emanations, researchers soon discovered that acoustic emanations, electromagnetic emanations and optical emanations are also potential sources of information leakage:

Acoustic emanation. Acoustic emanations from keyboards have been found to be excellent sources of disguised information. Since most encrypted systems utilize keyboards as input devices, acoustic emanations are compromising at best if reconstructed. An acoustic emanation can be decoded with feature matching using a single microphone (Marquardt et al., 2011) or with triangulation using multiple microphones (Fiona, 2006). Fortunately, the risk of acoustic eavesdropping decreases when the microphone is far away from the keyboard; with accuracy quickly dropping to random guesses at a distance of one foot.

Electromagnetic emanation. The innocuous seeming square waves and high switching frequencies are excellent radiation sources even up to the UHF region. The Electromagnetic spectra are rich with information and most of the means of communication irrespective of the form

reveal their little secrets with the EM spectra. This EM radiation is not merely interfering, but informative or compromising depending on one's attitude towards secrecy. Again, electromagnetic emanation is captured at close distances of less than one foot, with electromagnetic interference devices attached to the power line (Mo et al., 2013).

Optical emanation. In contrast to acoustic emanation, optical emanation focuses on commonly used output devices, *i.e.* the device's display. While there are multiple kinds of screens (e.g. CRT, LCD, LED), all of them allow the user to view content from certain angles. Using a camcorder within these angles will allow the adversary to acquire informative emanations. Unlike acoustic and electromagnetic emanation, optical emanation can be captured as far as 70 meters away (Xu et al., 2014a).

While optical emanation shows a superior attack range compared to acoustic and electromagnetic emanations, it is constrained by the viewing angle since visible light signals travel in straight lines. To overcome this limitation, reflections and effusions can be often utilized instead of observing the device screen directly.

2.4.1 Adversary Modeling

To better understand the problem and the challenges I face, I first analyze the properties of two specific reflection and effusion scenarios: (i) reconstructing typed input from repeated reflections; (ii) inferring TV content from light effusions. Although my adversary model is analyzed under such specific scenarios, it is worth noting that the analysis (*e.g.*, image size, noise, and impact of diffraction) can be generalized to similar situations.

2.4.1.1 Reconstructing Typed Input from Repeated Reflections

From the adversary's perspective, the techniques used to leverage so-called compromising reflections in prior work has inspired my awareness of the realism of these threats. However, they all suffer from a similar, and profound, weakness — the adversary must be either within visual

range of the victim (e.g., to ensure that pop-out events in reflections in the victim’s sunglasses can be discerned (Raguram et al., 2011)) or close enough to the target to avoid the use of expensive telescopes (Backes et al., 2009).

In my work, I consider a broader and more challenging adversary model: devices with any type of virtual or physical keyboard, without direct line-of-sight, at distances farther away from the victim than previously thought possible and even from reflections on the eye-ball. In these scenarios, the video captured by the adversary is noisy and the resolution is so low that visual cues such as the “pop-out” effect are no longer usable, completely defeating the existing attacks.

Captured Size of the Object. Obviously, the size of the object in the captured images is of critical importance and naturally depends on the size of the object itself. Other factors are the focal length and sensor resolution of the camera. Loosely speaking, the size of the object in the image for direct line-of-sight can be computed as:

$$Size_{Direct} = \underbrace{\frac{Sensor\ Resolution}{Sensor\ Size}}_{\text{pixel scale}} \cdot \underbrace{\frac{Object\ Size}{\frac{Target\ Distance}{Focal\ Length} - 1}}_{\text{size on sensor}} \quad (2.1)$$

Intuitively, the observed size is dependent on the physical size of the projection of the object on the sensor and the characteristics of the camera sensor, namely the size and number of pixels (picture elements). The size of the projection of the object on the sensor is controlled by the focal length, the distance to the object, and the object size. Focal length can be viewed as the amount of magnification of the image, where longer focal lengths (zoom lenses) provide higher magnifications. Thus, by using a lens with a longer focal length, an adversary can gain a better view of the target. The final size of the image of the device in pixels (given the image’s size on the sensor) then depends on the ratio between how many pixels are on the image sensor (*Sensor Resolution*) and the physical size of that sensor (*Sensor Size*). At the same focal length, for example, the size of the object in pixels tends to decrease with full frame sensors found in high-end digital SLRs compared to cheaper digital camcorders with smaller sensors but the same video resolution.

In addition to the physical object size, the size of the object on the image sensor also depends on the presence and shape of any reflecting surfaces between the observer and the object. For instance, if the reflecting object is convex (e.g., a pair of sunglasses or the human eyeball), the size of the observed object will be much smaller than if observed with direct line-of-sight. When an object is viewed via a reflection, the final observed size can be computed as:

$$Size_{Reflection} = Size_{Direct} * \frac{1}{\frac{2 \text{ Distance from Surface}}{Curvature Radius} + 1} \quad (2.2)$$

Thus, the curvature of the reflecting surface is an important factor in the observed size. The more curved the reflecting surface is, the more the light will be bent. For convex surfaces, the bending of the light will result in a smaller observed object size. Lastly, the distance between the reflecting surface and the target object itself (*Distance from Surface*) naturally affects the observed object size.

Takeaway. One way to acquire a larger observed size is to simply reduce the distance to the target object. However, from an adversarial point of view, it is desirable to be as far away as possible from the victim. Hence, a better solution would be to use a lens with a long focal length. For similar reasons, cameras with higher pixel density in their sensors are preferred. Finally, the curvature of any reflecting surface must be taken into account. For example, the human eyeball has a typical curvature of about 8 mm (Backes et al., 2008). Hence, when people look at an object 25 cm away, the reflection in their eyeball will be about 60 times smaller than with direct line-of-sight.

Impact of Diffraction. The quality of the acquired image is significantly influenced by the wave properties of the light. When light comes near the edge of the lens, not all light rays traveling from the object pass directly through the lens to the image sensor. Instead, a fraction of the light diffuses and travels in every direction, leading to a blurring of the image. This phenomenon is called diffraction and cannot be eliminated. It presents a physical boundary for the effective resolution of the object in the captured image (commonly referred to as the *Rayleigh Criterion* (Backes et al., 2008)).

The maximum effective size of the observed object (*Max Size*) can be approximated as:

$$Max\ Size = \frac{Aperture/Wavelength}{1.22\ Target\ Distance} * \frac{Object\ Size}{\frac{2Distance\ from\ Surface}{Curvature\ Radius} + 1} \quad (2.3)$$

Notice that the actual amount of diffraction is impacted by the wavelength of the light (*Wavelength*). While the adversary has no control over this factor, I include it here for completeness (along with the well-known constant scale factor of $1/1.22$ for circular apertures (Stelzer, 1998)). However, the adversary can select a lens with an appropriate aperture (i.e., opening of the lens), which lets the desired amount of unobstructed light pass through the lens.

Takeaway. The larger the aperture of the lens, the smaller the amount of diffraction. It is for precisely this reason that prior work (e.g., (Kuhn and Kuhn, 2003; Backes et al., 2008, 2009)) resorted to the use of telescopes. However, lenses with large apertures are typically very expensive, costing well over \$1,000 per square cm (Backes et al., 2008), and are difficult to conceal.

Impact of Noise. A very significant factor that affects the quality of the acquired image of the target object is imaging noise. Noise is a random variation in each pixel’s intensity, causing the image to appear speckled. As noted by Nakamura (2005), there can be several types of background noise in the captured image, each with a constant noise level. To avoid visual impact on the image quality by the noise, the exposure time is typically chosen so that the overall amount of light overwhelms the background noise, making it hardly noticeable. However, for video capture, the exposure is naturally limited to the time of a frame, which for darker scenes makes the background noise more noticeable. The resulting noise causes significant challenges in identifying fine detail.

Typically, cameras with large sensors are more resistant to noise, as their pixels are usually larger and can capture more photons of light. For that reason, the larger sensors provided in digital SLR cameras (as opposed to cellphones or point-and-shoot cameras) are desirable for photography even though they provide a smaller number of pixels on the object.

Taken as a whole, the aforementioned factors present challenges that *severely* limit the use of existing techniques (e.g., (Backes et al., 2008, 2009; Raguram et al., 2011, 2013)) when considering

reconstruction of typed input from repeated reflections. For instance, the recently used technique of identifying salient feature points (Raguram et al., 2011, 2013) within the image in order to facilitate alignment will fail because the poor image resolution does not provide the required details for the salient feature points. The approach suggested by (Maggi et al., 2011) faces similar challenges. Additionally, low image quality (e.g., as acquired from a reflection in the eyeball) prevents the detection of fine detail.

2.4.1.2 Inferring TV Content from Light Effusions

In this scenario, a potential adversary captures a video of the flickering light caused by changes in brightness on the device display and tries to confirm the video content being watched. This is a challenging task because the light signal from device display is weakened and distorted by reflections and effusions. The interference of nearby vehicle and road light might also be sources of noise. Because of heavy obfuscation, many of us may not have given a second thought to the amount of information these flickering patterns might reveal about the programs we watch.

The ability to confirm which video is being watched based off compromising diffusions of changes in light hinges on several factors, including (i) the quality of the captured information (i.e., the signal-to-noise ratio), (ii) the entropy of the observed information (i.e., the amount of variation in the captured signal), (iii) the length of the captured signal (e.g., short clips have more ambiguity), and (iv) the amount of information required for successfully matching the unknown and reference signals, which is related to the size of the adversary’s reference library and the distinctiveness of its contents I discuss each in turn.

Noise Interference. For an arbitrary recording, the adversary’s goal is to infer a signal, S , based on effusions of light from the display. In practice, this means that he or she also inadvertently captures an additive noise signal, N , which may be composed of a variety of other signals (e.g., sensor noise, photon noise). Consequently, the recording the adversary captures is the composition of the signal S and noise N . Intuitively, the more significant the noise, the harder it will be to distinguish between

the noise and the signal. This correlation is measured by the signal-to-noise ratio (SNR), which is the ratio of the signal variance σ_S^2 and the noise variance σ_N^2 .

In general, the higher the SNR, the less the noise influences the resulting signal, which leads to more robust signal analysis. In the case of capturing reflections of emanations, the SNR depends on a multitude of factors. More specifically, the amount of light emanated from the screen in any frame depends on the intensity of the video frame that is displayed on the screen, I_{ref} , the current brightness level of the screen (measured by unit area emanation power P_0), and the size of the screen S_{screen} . However, only a small fraction of this light might be captured by the camera, the amount of which depends on the distance the light travels from the screen to the reflecting object, the size and reflectance of the reflecting object, the aperture of the camera and the distance from the reflecting object to the camera. The captured signal also depends on the sensitivity α_{cam} of the imaging sensor of the recording device. In summary, assuming α_{cap} is the percentage of emanation captured by the camera, the recorded signal can be modeled as:

$$I_{cap} = I_{ref} P_0 S_{screen} \alpha_{cap} \alpha_{cam} \quad (2.4)$$

It is important to note that α_{cap} and α_{cam} are not constant in practice because of the different reflectance properties for different colors and the non-linear color transformations of digital cameras (Tsin et al., 2001). Hence, they will depend on the actual color composition of the displayed video frame. Additionally, the intensity of light in the room influences the amount of incoming light and could be treated as another signal, but for simplicity, I consider it an additive constant (which assumes the lights are not being repeatedly turned on and off). As such, I omit its embedded signal in Equation 2.4, but instead simply consider it as a source of “impulse noise” (Buades et al., 2005), similar to the lights of a passing vehicle.

To complicate matters even further, there can be noise from a myriad of other sources that impact the measured brightness value in the adversary’s recording of the emanations coming from the display. These include quantization noise of the camera during the A/D conversion to

obtain pixel values (Widrow and Kollár, 2008), thermal noise from the sensor itself (Healey and Kondepudy, 1994), and impulse noise. Again, for simplicity, I accumulate the above noise factors into a single noise variable I_{noise} . The SNR can then be computed as:

$$SNR = \frac{\sigma(I_{cap})^2}{\sigma(I_{noise})^2} \quad (2.5)$$

where $\sigma^2(\cdot)$ is the variance of the signal.

Intuitively, lower screen brightness levels, smaller and darker reflecting objects, and longer distances limit the amount screen light captured by the adversary. Fortunately for the adversary, a high quality camera can capture a good percentage of the incoming light and reduce quantization and electronic noise. Finally, note that while the intensity of a constant room light does not influence the SNR directly—since it does not influence the noise variance—it indirectly effects the quantization noise given that it affects the sensitivity of camera’s sensor (i.e., higher room light intensity makes the camera less able to capture subtle illumination changes).

For my experimental evaluations I can directly acquire $I_{cap} + I_{noise}$ from the captured video. An estimate of the noise variance $\sigma(I_{noise})^2$ can be measured by having the adversary capture the reflection from a static image displayed on screen beforehand (e.g., at her house). Similarly, room brightness can be approximated. With these measurements at hand, I_{cap} can be estimated with linear regression using I_{ref} , and the SNR can be directly computed.

Takeaway. The factors that influence the signal I am interested in can be approximated by Equation 2.4. Moreover, by using Equation 2.5, I can infer the SNR directly from the captured data, which ranged from 5 to 107 in my empirical evaluations.

Point of Capture. Obviously, the point from which the light diffusions are recorded influences how well the attacker can confirm her hypotheses. Intuitively, the more she is able to record sudden intensity changes, the higher the chances that the correct content will be inferred. A key challenge for the adversary is that the average intensity of one frame is highly dependent on that of the

previous frame. For instance, in my empirical evaluations, nearly 95% of consecutive frames have the same average intensity (up to rounding error precision).

To improve my ability to carry out the attack, I do not use the raw data directly, but instead use its gradient to reduce the correlation. To see why that helps, assume that $x_t = I_{ref}(t+1) - I_{ref}(t)$, $y_t = I_{cap}(t+1) - I_{cap}(t)$, $t = 1, 2, 3, \dots$. Then, given that the vast majority (i.e., 95%) of the average intensities are similar, this means that 95% of the x s would be 0. Assuming the gradients are independent of one another, the information in a particular frame sequence $\{x_t | t = 0, 1, 2, 3, \dots\}$ can be measured as:

$$I_{ref}(x) = -\sum_{t=0}^N \log(f(x_t) \Delta x) \quad (2.6)$$

where $f(x_t)$ is the probability density function (PDF) of x_t in a single frame. $I_{ref}(x)$ can be viewed as the logarithm of the inverse probability of the reference sequence. The higher its value, the less likely another reference sequence will “accidentally” be the same as it, which means that the sequence has less ambiguity and contains more information. Consequently, the more intensity changes the adversary observes, the more likely it is that the correct content will be inferred.

To gauge how well the attack would work, I can compute the mutual information between x and y using Equation 2.7. $I_{mutual}(x, y)$ estimates the information captured by the adversary on average.

$$I_{mutual}(x, y) = \int p(x)p(y|x) \log\left(\frac{p(y|x)}{p(y)}\right) dx dy \quad (2.7)$$

In practice, $p(x)$ can be observed directly from a reference video. Likewise, $p(y)$ can be computed by ignoring impact noise (which is rare) and assuming that the noise follows a Gaussian distribution. In fact, since I_{ref} and I_{cap} are linearly related, I can also assume $y = x + noise$, where $Var\{noise\} = Var\{x\}/SNR$. In doing so, I can now compute the mutual information with the SNR I acquired. For context, I note that in my evaluations that follow, at an SNR of 5, every frame conveyed roughly 1.5 bits of information. Under much better conditions with an SNR of 107

(observed when the diffusions were captured while the victim watched an action scene on a 50-inch TV), every frame conveyed 3.2 bits of information.

Takeaway. The above analysis tells us what one would expect: the more intensity changes that we observed, the less the resulting ambiguity. Therefore, if the adversary is lucky enough to observe several sharp changes in intensity, she will have an easier time to identify the content being watched by the victim. Not surprisingly, Equation 2.7 also tells us that bigger and brighter screens provide more than twice as much information (compared to the smaller and darker ones used in my experiments).

Length of Recording. Given the previous discussions, longer recordings are obviously better for the adversary. To see that, assume that the arrival of intensity changes are Markov, meaning that the distribution of arrival time and magnitude of the next intensity change depends only on the current state of the video being watched. If that is the case, then the information learned by the adversary is linearly related to the mutual information per frame. Ideally, the attacker’s best hope is for a high SNR environment, a good starting point, and a suitable recording length capturing multiple changes in intensity.

Size of the Reference Library. The last factor that affects the speed and accuracy of the attack is the size of the reference collection the adversary must test her hypotheses against. On average, the amount of information I need to uniquely identify a video is logarithmic with its total length, which in turn, is linearly related to the size of the attacker’s library. Therefore, linearly increasing the size of the library will only have marginal influence on her ability to successfully confirm which content the victim is watching.

2.4.2 Most Related Works

In this section, I investigate in detail several of the most related works in both the security and visual recognition and retrieval domains.

2.4.2.1 Related Work in Security

In modern computer security research, probably the earliest idea of compromising emanation is embodied in the work of van Eck (1985) and Highland (1986) on compromising signals from electromagnetic radiation. That work was later extended by Kuhn and Kuhn (2003), wherein it was argued that a telescope could be used to spy on reflections of a computer screen from afar. Intuitively, the light emitted from the display was modeled as a function of time and space, and then analyzed using signal processing methods. Naturally, the captured light is a disturbed signal of the light emitted by the screen, where disturbances include atmospheric effects, dispersion, attenuation, and lens distortion of the capture device. Regardless of these disturbances, Kuhn and Kuhn (2003) show that by utilizing an expensive telescope with a wide aperture, they were able to reconstruct text on a 32×24 cm display from 60 m away.

More recently, Backes *et al.* (Backes et al., 2008, 2009) overcame the requirement of direct line-of-sight. The key innovation was to exploit reflections to vary the path of light between the target screen and the observer, and showed it was possible to leverage the reflection off a human eyeball (reading a very large 36 pt text font from a distance of 10 m). However, the approach still used a high-powered telescope,, inheriting the drawbacks of high cost along with limited versatility and ability to go undetected. In addition, the setting of Backes *et al.* did not have to consider motion (of either the victim’s device or the adversary), and also was not concerned with the daunting task of automatically reconstructing text from typed input.

The use of less expensive and more practical equipment was introduced by Raguram et al. (2011). Unlike the approach I present, that method relied on detecting the presence of key pop-outs in virtual keyboards. While that approach worked well for direct line-of-sight attacks, reconstructions involving a single reflection (in this case, off the victim’s sunglasses) did not perform as well (Raguram et al., 2011). A related approach that also relied on the ability to detect pop-outs was proposed by Maggi et al. (2011). However, the approach of Maggi et al. is sensitive to movement of the device and camera and suffers in the presence of occlusions (including the fingers of the device’s

user). Neither approach could handle reconstructions of low resolution images of reflections of reflections in nearby objects.

Later, Yue et al. (2014) and Shukla et al. (2014) proposed two different approaches that utilize finger movement and microscopic frame level dynamics to reconstruct typing behavior. Unlike previous attacks, their approaches do not require a view of the device display, but solely use finger movements as the data source. Yue et al. (2014)’s approach focuses on the movement of fingertips and identifying key pressing behavior when the fingertip touches the device screen. Shukla et al. (2014), on the other hand, detect typing behavior using the relative distance between feature points on the user’s hand and device. The temporal changes of the relative distance reveals the timing of typing behavior while the magnitude of relative distance reveals the key being typed. Yet both of the attacks require direct a view of the victim.

In a similar manner, Torralba and Freeman (2012) make use of reflections to reveal “accidental” scenes from within a still image or video sequence. The advantage for these approaches comes from the uniformity and easy-access of visual signals; while TV and computer screens come with different models using different technologies — resulting in extremely different electromagnetic behavior — they all share similar visual output. Due to market demand, the visual signals have to cover a certain area and maintain a certain brightness level to ensure clarity of picture, which also make them susceptible to compromising reflections. That said, these attacks require a view of the screen, either directly or via reflections.

Lastly, Enev et al. (2011) and Greveler et al. (2012) proposed techniques for undermining a user’s privacy via TV program retrieval, wherein power usage and power line electromagnetic interference were investigated as side-channels. This work infers the TV signal in ways that largely depend on the model of the TV and the structure of the power system. Therefore, to successfully carry out the attack, an adversary must not only have specialized equipment and access to smart electrical meters, but must also have a priori knowledge of the victim’s TV model — all of which weaken the practicality of the attack. Moreover, other electronic devices (e.g., computers) within

the vicinity of the TV can interfere with the captured signal, compounding the decoding challenges even further.

2.4.2.2 Related Work in Recognition and Retrieval

Within the computer vision and human computer interaction communities, work on finger tracking for gesture recognition (Oka et al., 2002; Chaudhary et al., 2012), virtual input systems (Ukita and Kidode, 2004; Zhang, 2003), virtual object manipulation (Lee and Hollerer, 2007; Lee and Chun, 2009), and hand writing recognition (Yang et al., 2005; Jin et al., 2007) all share similarities to my application of finger motion analysis. Probably the most germane of these is the work of Iturbe et al. (2008) which uses finger movement to control a virtual input system in which a user's finger is modeled as a straight line and its movement is used to determine the button being pointed to by the user. Unfortunately, their approach quickly fails for small mobile devices where the fingers need to bend in order to reach the keys. Similarly, Jin et al. (2007) utilized finger input for character writing recognition. In their approach, the user's finger is isolated using a background modeling technique. Next, the path taken by the finger is tracked as the user writes individual letters, effectively recognizing the letter through the trajectory of the finger. Unfortunately, the approach of Jin et al. (2007) cannot be directly applied to mobile devices as users do not spell words by forming one character at a time, but instead interact with a keyboard via a series of touch events.

Kerdvibulvech and Saito (2007) apply a novel technique for tracking the fingers while a user plays a guitar. Instead of trying to uniquely identify each individual finger, the authors use a neural network classifier to recognize known patterns corresponding to different chord formations. While promising, their approach is also not applicable in my setting, as the way users type on mobile devices can differ significantly for the same user (e.g., switching between typing with one, two, or several fingers), and even more among different users, making the learning strategy less practical. Nevertheless, as I show later, I found that by combining many of the strengths of prior works alongside my own enhancements, I am able to achieve a solution that surpasses previous attempts in terms of its practicality, range, and robustness.

Also related within the domain of computer vision is the process of image and video retrieval. Interested readers are referred to Zhang and Rui (2013), which presents an excellent review of image retrieval techniques used to search through billions of images. Likewise, Liu et al. (2013) presents a survey of near-duplicated video retrieval techniques that also focuses on similarity of semantic content of the video sequences. In short, features are extracted to reveal detailed information in the image and semantic labels are used to provide a high level understanding. Unfortunately, I have no such luxury in my application since I may have no visual access either directly or indirectly to the screen, and must therefore find ways to work with much more limited information.

Lastly, my application domain shares similarities to genome sequence matching and database searching. In particular, considering only the average image intensity signal, the task at hand can be viewed as a sequence matching problem. For instance, in genome sequence matching, Langmead et al. (2009) present a fast DNA sequence matching scheme that exploits time and space trade-offs. In database searching, Faloutsos et al. (1994) and Moon et al. (2002) present methods that perform fast matching from an input subsequence to those in a database. Unfortunately, these techniques suffer from several limitations that make them ill-suited for my setting. For example, in DNA sequence matching, many parts of a sequence may be missing and so to find the best matches, dynamic optimization methods are usually deployed to maximize the length of the best match. These algorithms typically have $O(mn)$ complexity, where m is the length of the query sequence and n is the length of the reference sequence. In my application, however, the only uncertainty is the starting point of the query sequence and so much more effective strategies (i.e., $O(n\log(m))$ or faster) can be applied.

In database searching, the problem is more similar to mine, but the state-of-the-art solutions utilize Fourier transformation and focus on low frequencies. In my application, the sudden intensity changes contain most of the information I utilize, but live in the high end of the frequency spectrum. As such, these approaches cannot be directly applied. However, by combining many of the strengths of prior work together with my own enhancements, I provide a solution that boasts high accuracy and speed.

2.4.3 Remaining Challenges

While the use of off-the-self video cameras to perform optical attacks has been investigated to varying degrees, the practicality of these attacks has not been well studied. Intuitively, a long range attack is much easier to conceal than a close one. My approach deals with this challenge and significantly pushes forward the upper limit of the attack range.

Specifically, I propose automated optical emanation compromising attacks that utilize reflections and effusions, which can be performed at a much longer range than previous work. To make this happen, I propose innovative computer vision approaches in video event detection and retrieval that are robust against noise, low resolution, and low brightness level.

2.5 Content-Based Copy Detection (CBCD)

While the techniques used by online social networks to detect pirated content have been successful, they have been widely criticized for false detections (Bartholomew, 2014)¹. Improving the accuracy of such systems remains an important social problem and immense business opportunity, and as such, has garnered much attention from both academia and industry. However, content-based copy detection — especially as it pertains to *subsequence similarity* — is more difficult than it might appear at first blush. This is particularly true when transformations to the original content might be made either in video or in audio², either spatially or temporally, and may be either simple or compound. So pressing is the problem of content identification that in 2008, the U.S. National Institute of Standards and Technology (NIST) included a separate challenge on video copy detection in their annual TREC Video Retrieval Evaluation (TRECVID). In that challenge, synthetic video-based transformations were used to test the performance of different content detection approaches.

¹ See also P. Tassi, “*The Injustice of the YouTube Content ID Crackdown Reveals Google’s Dark Side*” Forbes Magazine, 2013.

² See S. Smitelli “*Fun with YouTube’s Audio Content ID System*”, available at <http://www.scottsmitelli.com/articles/youtube-audio-content-id>

Four years later, the challenge was prematurely terminated, claiming that near-duplicate video detection was a solved problem. However, in 2014, Jiang et al. (2014) showed that the current state of the art — which showed near-perfect results on the simulated benchmarks — is far from satisfactory in detecting complex real-world copies. To highlight the problem, Jiang et al. (2014) released a new dataset (called VCDB) that contains pirated videos available on YouTube and MetaCafe. Their preliminary evaluations suggest that the transformations observed in the real world are very different from the synthetic transformations considered by the academic community to date. For instance, the most widely studied transformation (i.e., “picture in picture”) in the TRECVID evaluations is rarely seen in real cases, while the more commonly used transformations observed online are far more complex than the synthetic ones tested in past works. As a result, the techniques that appear to be robust in simulated benchmarks fail miserably in the wild. Jiang et al. (2014) confirmed my own suspicion that the adversarial assumptions made in the NIST challenges were far too naïve and simply did not capture the behavior of real-world adversaries.

To better understand the problem and challenges, I first analyze the adversary model in realistic copy detection scenarios and then review related work in this domain.

2.5.1 Adversary Modeling

To better identify the challenges in this scenario, I first explored several widely used sources of data. I turned to the multimedia community, where numerous datasets for evaluating copy detection technologies are publicly available. The most widely cited of these are the MUSCLE-VCD-2007 dataset (with over 100 hours of video) (Law-To et al., 2007) and the IACC dataset (with over 200 hours of video) released by NIST for the TRECVID challenge (Smeaton et al., 2006). Both datasets contain material from a wide variety of sources and also include simulated “pirated” videos wherein various synthetic transformations were applied. These transformations were either applied to short segments within a video or to the entire video. The duration of a copied segment ranged from 1 second to 1 minute.

Unfortunately, upon close inspection of these datasets, I was surprised to learn how naïve the transformations were (with little, or no, temporal adjustments involved). Fortunately, recently released VCDB dataset published by Jiang et al. (2014) contains 528 videos retrieved from two user generated content (UGC) platforms using 28 queries. Among those videos, the authors manually labelled 9,236 pairs of copied segments. Examples of transformations observed in the VCDB dataset are shown in Figure 2.1.

Compared with previously used synthetic datasets, the VCDB dataset is extracted and labelled directly from real data. Although it suffers from the limited sample size, the VCDB dataset consists the most comprehensive modeling of the adversary to my best knowledge. Therefore, I analyze the properties of the VCDB dataset as my target adversary model.



Figure 2.1: Copy-reference pairs in the VCDB dataset

To get a better sense of the perceived quality of both the synthetic and real-world transformed videos, I investigate to objective measures for video quality assessment (Chikkerur et al., 2011). While most work in that domain only considers spatial distortions due to network coding, Soundararajan and Bovik (2013) suggest an approach (coined the RRED index) that also assesses temporal distortions. At a high level, the RRED index computes the mutual information in both the spatial frequency and temporal frequency domains, and the amount of information between the distorted video and reference video yields a score depicting how much distortion is involved. One would expect that the higher the distortion, the more likely that viewers would find the transformed

Video Source	Copy-Reference Pair from IACC	Copy-Reference Pair from VCDB	Two Random Videos
25th Percentile	118.5	56.8	141.1
Median	217.8	215.7	244.2
75th Percentile	381.6	320.5	542.0

Table 2.1: RRED index score statistics for copy-reference pairs in the IACC and VCDB dataset, compared with the result from random videos

video distracting. The experimental analysis of Soundararajan and Bovik (2013) indicates that their approach provides scores that correlate well with human judgements. Unfortunately, my analysis shows that the RRED index does not adequately capture the impact of distortions besides noise. To see that, Table 2.1 shows the RRED index scores for copy-reference pairs in the IACC and VCDB datasets. Keep in mind that the analysis of Jiang et al. (2014) aptly showed that all the CBCD techniques that achieved near perfect scores on the IACC dataset failed miserably on the VCDB dataset. Notice, however, that although slightly lower scores are assigned for two similar copy-reference pairs than for two different videos, the range of scores overlaps significantly. Thus, left with no other way to objectively measure the quality of the transformations, I use the popularity of the pirated videos in the VCDB dataset as an indication of the perceived quality of the transformations, the fact that users were willing to watch the pirated content even with these transformations should not be overlooked. Figure 2.2 shows that most of the pirated videos in the dataset were watched on YouTube more than 300 thousand times.³

2.5.1.1 Findings

Having acquired copies of pirated content observed on UGC platforms, I began a thorough analysis of the statistics of the data in order to ground my adversarial assumptions. Although the VCDB dataset provided loose labelling for the copy-reference pairs, to fully understand the data, frame-level labelling was required. To accomplish that goal, I built a graphical user interface for labelling corresponding frames in the videos and manually labelled a subset of 210 videos that were

³ Recall that even though the videos still appear on YouTube, it is not clear if they were missed by Content ID or if they were flagged but the copyright owner choose to redirect all monetization and ad revenue instead of having the infringing videos removed.

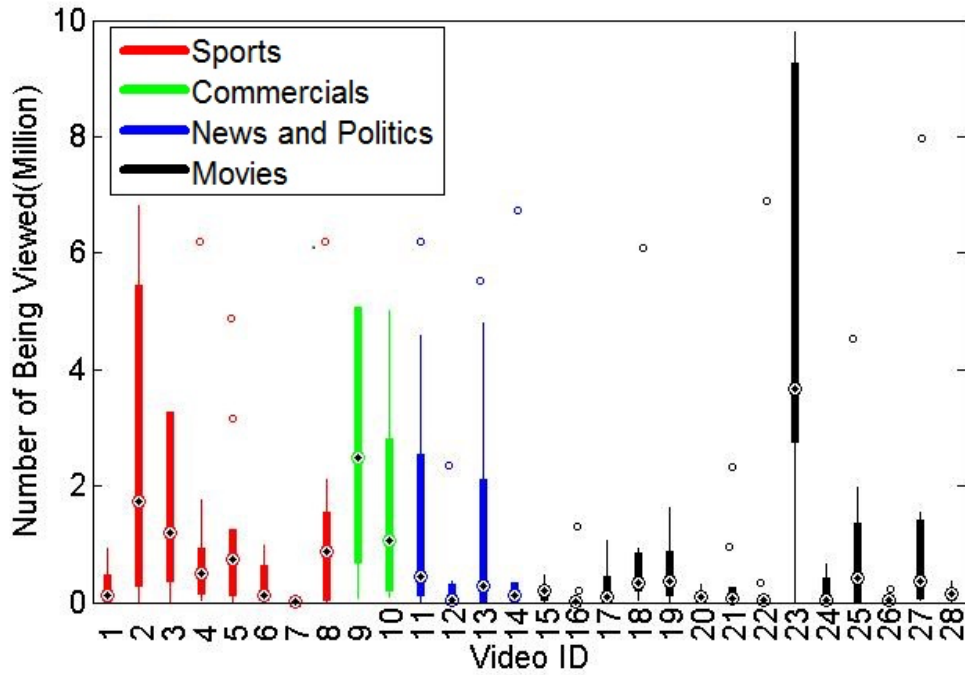


Figure 2.2: Viewership statistics for the videos in the VCDB dataset

all over 30 seconds long. Among those videos, I found 4402 copy-reference segment pairs, all of which were manually verified.

The videos from the VCDB dataset contain a wide range of content transformations, which is in stark contrast to the pre-defined lab-generated transformations (Smeaton et al., 2006). For instance, 8% of copies were edited to have parts of the original video deleted or extra segments inserted. The distribution of transformations is also different from that in IACC. Among the 9,236 pairs of copies, 36% contain insertion of patterns, 18% are due to camcording, 27% have scale changes, and only 2% contain “picture in picture” patterns. Digging deeper, I focused on the temporal characteristic of the transformed videos. Two types of temporal transformations were immediately apparent: speed adjustment and segment editing (i.e., video cropping and temporal reversing). I discuss each in turn.

Speed Adjustments: To characterize the observed video speed transformations, I computed the distribution of video speed. Instead of observing video speed directly, I use the logarithm of the video speed ratio (i.e., speed of the distorted video divided by the speed of the original video).

This distribution is well modeled by a log-logistic distribution. I observed only 44.2% of pirated videos have no temporal scaling and only 2.2% are scaled by more than 20%. Next, to analyze the consistency of the speed adjustments, I computed the standard deviation of the video speed within each pirated content. The results (not shown) revealed that as many as 97.7% of the pirated videos do not have different speeds within the video.

Cropping, Editing and Reversing: My analysis of the data collected from the UGC platforms indicated that roughly 40% of the videos contained more than 80% content copied from elsewhere. These results are consistent with the observations of Jiang et al. (2014). Over 8% of the samples contain edited video segments wherein segments were either deleted or inserted. 2% of the videos were temporally reversed. The low percentage of temporal reversion makes sense because few videos are still watchable once reversed.

2.5.1.2 Assumptions

The limited types of temporal transformations observed in the real-world data suggest that although the magnitude of the intensity signal in a copied video may vary a lot due to spatial transformations, the temporal positions of sudden intensity changes remain relatively constant. I assume that to be true because harsh temporal adjustments appear to degrade the viewing experience more than similarly strong spatial adjustments (e.g., degraded video quality).

2.5.2 Related Work

Although there is abundant literature on CBCD approaches, all of them share the same detection structure: feature extraction and then indexing.

The feature extraction phase identifies fingerprints of both the query video and the reference videos. It also defines the measure to compute the difference between two fingerprints. A successful feature extraction scheme should be robust against transformations. Namely, the fingerprints from a transformed video and a reference video should result in a close distance measure, compared with non-reference videos.

The indexing phase efficiently compares the query video against the dataset. The naive way is to run linear searching within the dataset. However, given the size of datasets, this is usually not practical. To solve this problem, data structures such as Kd-tree and hashing tables are used for efficiency.

Since feature extraction is essential to the accuracy and success rate of video copy detection, I mainly focus my survey on this area.

2.5.2.1 Spatial Features

The most popular features used in CBCD are spatial features. These features characterize the property of individual frames and can be divided into global features and local features.

Global Features. The most well known global feature is the ordinal feature (Hampapur et al., 2001) (Hua et al., 2004). In this approach, as image frame is divided into K blocks. The ordinal feature is defined as the K ranking of the average intensity value of each block.

Other similar global spatial features include spatial correlation descriptor (Yeh and Cheng, 2009), MPEG-7 descriptors (Küçükünç et al., 2010), and TIRI descriptor (Esmaeili et al., 2011). The advantage of the spatial global feature is as follows. First, global features are more robust against noise and global changes introduced by the digitization/encoding process. In Arun Hampapur et. al's approach, the ordinal feature is compared against motion signature and color signature. The ordinal feature showed superior results in detecting short video clips and computation efficiency. In addition, unlike local features, spatial global features usually result in relatively low dimension data, which makes feature matching computationally efficient.

However, spatial global feature such as ordinal feature is not robust against image shift, adding black margins or black borders and flipping.

Local Features. The most widely used local feature includes Harris corner features (Joly et al., 2007) (Law-To et al., 2006) (Law-To et al., 2009) (Pouillot et al., 2007), SIFT/SURF feature

(Gengembre et al., 2008) (Ke et al., 2004) (Zhang and Chang, 2004), and their variations such as Hessian-based STIP descriptor (Willems et al., 2008). Local features extract visually significant points in the image (corners, spots, tips, etc.). Every local feature only represents the texture of a small region. To fully encode one image, these approaches extract hundreds of local features in each single frame.

The use of local features provides detailed information about every frame in the video. This makes it possible to retrieve much shorter copy sequences. Moreover, they are robust against image shift, scaling and rotation. However, local features are sensitive to noise and blurring. They are also computationally expensive for efficient matching.

The most traditional measure for spatial features is through voting (Law-To et al., 2006). Reference videos in the dataset vote for how many similar spatial features are detected in the test video. However, this scheme completely ignores the relation skip between frames, not taking advantage of temporal information.

More recently, researchers have proposed to use more complicated schemes for difference measuring so that the temporal relationship between frames can be identified. More sophisticated models were proposed to take temporal adjustment (frame dropping, speed changing, etc.) into consideration. These approaches include frame fusion (Wei et al., 2011), bag-of-words model (Chiu et al., 2007) and graph based matching (Chiu et al., 2008). These models takes temporal relationship between frames into consideration and work with temporal transformation such as speed adjustment.

However, global spatial features are sensitive to bordering and shifting, while local features are sensitive to noise. These sensitivity makes spatial-based CBCD detections always vulnerable to some kind of spatial transformations. Moreover, although spatial methods may require only short sequences for detection, the complexity of the features makes these schemes less computationally efficient. If the user cares more about long copies (longer than 1 minute), temporal features might provide easier and more efficient retrieval results.

2.5.2.2 Temporal Features

Instead of focusing on individual frames, temporal features characterize the changes of pixel values along time. However, a very limited number of papers have worked on this domain.

The most relevant approach is Jean-Hugues Chenot et. al. (Chenot and Daigneault, 2014), which designed a temporal fingerprint based on spectral analysis of intensity changes. It was the first attempt of CBCD to characterize the temporal behaviour of the video. Since the fingerprint only relies on average intensity changes, this approach is resistant against typical spacial transformations, but is very sensitive to temporal adjustments. Moreover, since only the first 16 channels of the FFT transformation are used as fingerprint, this approach fails to characterize sudden illumination changes that contain the most information.

Another related approach is from (Chen and Stentiford, 2008), which applied ordinal measure in the temporal domain on different blocks of the image. While the ordinal measurement captures temporal characteristics, tuning the number of blocks enables the algorithm to seek a balance between spatial robustness and detection efficiency. However, this approach fails to characterize the magnitude of temporal changes. It is also sensitive to temporal adjustments such as speed changes, temporal smoothing and frame dropping.

In my latest work (Xu et al., 2014a), I also propose a TV content retrieval scheme based on temporal information. Different from previous research, my approach focuses on the timing of sudden illumination changes, since they contain the most of the information. Similar with previous approaches, my method is robust against spatial changes. It also shares the disadvantage of sensitivity against temporal adjustments, but the discrete nature of sudden illumination changes makes it more promising to be temporally robust with some improvements.

2.5.2.3 Indexing

Different indexing methods also play an important role in efficient copy detection. The most popular indexing approaches involve local sensitive hashing and tree-based searching (R-tree, Kd-

Video Source	Copy-Reference Pair from IACC dataset	Copy-Reference Pair from VCDB dataset	Random Two Different Videos
25% Percentile	118.5	56.8	141.1
Median%	217.8	215.7	244.2
75% Percentile%	381.6	320.5	542.0

Table 2.2: RRED index score statistics for copy-reference pairs in IACC and VCDB dataset, compared with result from random different videos

tree, K-means tree or B^+ -tree) (Shen et al., 2005) (Yuan et al., 2004). In my latest work (Xu et al., 2014a), I also built a Kd-tree structure for efficient retrieval.

2.5.2.4 Video Quality Assessment

Additionally, I introduce another set of papers that is relevant to the topic: full reference and part reference video quality assessment (Chikkerur et al., 2011). Video quality assessment evaluates how a video is distorted from a human perspective. It mimics the human vision system to determine how much video distortion is involved. Video quality assessment mainly considers spatial distortions from network coding and transferring.

A most relevant approach considering both spatial and temporal distortion is the RRED index (Soundararajan and Bovik, 2013). Their work computes the mutual information in both spatial frequency and temporal frequency domains. The amount of mutual information between distorted video and reference video provides a score revealing how much distortion is involved. The experiments indicate that their result has high correlation with human judgement.

However, video quality assessment approaches fail to take the variety of distortions into consideration. When I apply RRED index to IACC dataset and VCDB dataset, they fail miserably, failing to assess copies with transformations other than adding noise. The results are listed in Table 2.2, indicating that RRED index results in slightly lower scores for copy-reference pairs than two different videos but the score intervals are highly overlapped. It only results in 12% and 22% precision with 100% recall rate. This might be useful to select videos within minor distortions, but fail on complicated transformations.

2.5.2.5 Audio Matching

Lastly, sequence matching is also a widely discussed topic in audio retrieval and speech recognition. The standard approach for these tasks is to represent the audio sequence in the frequency domain and handle temporal transformations through dynamic temporal warping (DTW). For instance, Raffel and Ellis (Raffel and Ellis, 2015) represent query audio data with constant-Q transforms (CQT) and align them against reference sequences with DTW. In speech recognition, a similar use of DTW was proposed by Godin and Lockwood (Godin and Lockwood, 1989). DTW breaks the audio input sequences into discrete elements, which can then be matched in a manner that is robust against non-linear transformations. The usage of DTW in audio matching is guided by the fact that the frequency property of a music note or a spoken word is highly informative and relatively consistent against temporal transformation. However, DTW is not well applicable in my CBCD scenario as the spatial features are sensitive to spatial transformations, and cannot be treated as robust DTW elements. Additionally, the temporal sampling rate of visual data is only 30 Hz compared to at least 8,000 Hz for audio data, making visual data much less informative in the temporal domain. Therefore, it is difficult to extract informative temporal features as DTW elements from visual data. My proposed methods overcome these limitations and achieve robust matching under spatial and temporal transformations. Before introducing my approach, I model the adversary setting with statistical results.

In what follows, I delve into details for each of the aforementioned scenarios.

CHAPTER 3: DECODING CODEWORDS FROM MOTION-BASED CAPTCHAS

3.1 Introduction

Humans can recognize a wide variety of objects at a glance, with no apparent effort, despite tremendous variations in the appearance of visual objects, and we can answer a variety of questions regarding shape properties and spatial relationships of what we see. The apparent ease with which we recognize objects belies the magnitude of this feat. we can also do so with astonishing speed (e.g., in a fraction of a second) (Thorpe et al., 1996). Indeed, the Cognitive Science literature abounds with studies on visual perception showing that, for the most part, people do not require noticeably more processing time for object categorization (e.g., deciding whether the object is a bird, a flower, a car) than for more fine grained object classification (e.g., an eagle, a rose) (DiCarlo and Cox, 2007). Grill et al. (Grill-Spector and Kanwisher, 2005) showed that by the time subjects knew that a picture contained an object at all, they already knew its class. If such easy-for-human tasks are, in contrast, difficult for computers, then they are strong candidates for distinguishing humans from machines.

Since understanding what we see requires *cognitive* ability, it is unsurprising that the decoding of motion-based challenges has been adopted as a security mechanism: various forms of motion-based object recognition tasks have been suggested as reverse Turing tests, or what are called Completely Automated Public Turing tests to tell Computers and Humans Apart (CAPTCHAs). Among the key properties of CAPTCHAs are: they must be easily solved by humans; they should be usable; correct solutions should only be attainable by solving the underlying AI problem they are based on; they should be robust (i.e., resist automated attacks); and the cost of answering challenges with automated programs should exceed that of soliciting humans to do the same task (Ahn et al., 2003; von Ahn et al., 2004). To date, a myriad of text, audio, and video-based CAPTCHAs have been

suggested (Hidalgo and Alvarez, 2011), many of which have succumbed to different attacks (Golle, 2008; Mori and Malik, 2003; Yan and Ahmad, 2007, 2008a; Zhu et al., 2010; Bursztein et al., 2011b,a).

While text-based CAPTCHAs that prompt users to recognize distorted characters have been the most popular form to date, motion-based or video CAPTCHAs that provide some form of moving challenge have recently been proposed as the successor to static CAPTCHAs. One prominent and contemporary example of this new breed of CAPTCHAs is NuCaptcha (NuCaptcha, 2011), which asserts to be “*the most secure and usable CAPTCHA*,” and serves millions of video CAPTCHAs per day. The general idea embodied in these approaches is to exploit the remarkable perceptual abilities of humans to unravel structure-from-motion (Marr and Poggio, 1979). For example, users are shown a video with a series of characters (so-called random *codewords*) moving across a dynamic scene, and solve the CAPTCHA by entering the correct codeword. For enhanced security, the codewords are presented among adversarial clutter (Mori and Malik, 2003) (e.g., moving backgrounds and other objects with different trajectories), and consecutive characters may even overlap significantly. The underlying assumption is that attacks based on state-of-the-art computer vision techniques are likely to fail at uncovering these challenges within video sequences, whereas real users will be able to solve the challenges with little effort.

However, unlike in humans, it turns out that object *classification*, not recognition of known objects, is the more challenging problem in Computer Vision (Ullman, 2000). That is, it is considerably more difficult to capture in a computer recognition system the essence of a dog, a horse, or a tree—i.e., the kind of classification that is natural and immediate for the human visual system (Marr, 1982). To this day, classification of objects in real-world scenes remains an open and difficult problem. Recognizing known objects, on the other hand, is more tractable, especially where it involves specific shapes undergoing transformations that are easy to compensate for. As I show later, many of these well-defined transformations hold in current motion-based CAPTCHA designs, due in part to design choices that increase usability.

In what follows, I present an automated attack to defeat the current state-of-the-art in moving-image object recognition CAPTCHAs. Through extensive evaluation of several thousand real-world CAPTCHAs, my attack can completely undermine the security of the most prominent examples of these, namely those currently generated by NuCaptcha. After examining properties that enable my attack, I explore a series of security countermeasures designed to reduce the success of my attacks, including natural extensions to the scheme under examination, as well as an implementation of a recently proposed idea (called Emerging Images (Mitra et al., 2009)) for which attacks do not appear as readily available. Rather than idle conjecture about the efficacy of countermeasures, I implement CAPTCHAs implementing them and evaluate these strengthened variations of moving-image CAPTCHAs by carrying out and reporting on a usability study with subjects asked to solve such CAPTCHAs.

My findings highlight the well-known tension between security and usability, which often have subtle influences on each other. In particular, I show that the design of robust and usable moving-image CAPTCHAs is much harder than it looks. For example, while such CAPTCHAs may be more usable than their still-based counterparts, they provide an attacker with a significant number of views of the target, each providing opportunities to increase the confidence of guesses. Thus the challenge is limiting the volume of visual cues available to automated attacks, without adversely impacting usability.

3.2 Our Automated Approach

The human visual processes of segmentation, object tracking, and region identification are possible in today’s MIOR CAPTCHAs because of several design decisions that promote rapid visual identification (Driver and Baylis, 1996). NuCaptcha, for instance, presents a streaming video containing moving text against a dynamic background. The videos have four noticeable characteristics, namely: (1) the letters are presented as rigid objects in order to improve a user’s ability to recognize the characters; (2) the background video and the foreground character color are nearly constant in color and always maintain a high contrast—we posit that this is done to ease



Figure 3.1: Example moving-image object recognition (MIOR) CAPTCHAs from NuCaptcha (see <http://nucaptcha.com/demo>).

cognitive burden on users; (3) the random “codewords” each have independent (but overlapping trajectories) which better enable users to distinguish adjacent characters; (4) lastly, the codewords are chosen from a reduced alphabet where easily confused characters are omitted. Some examples of a state-of-the-art MIOR CAPTCHA are given in Figure 3.1.

Before delving into the specifics of my most successful attack, I first present a naïve approach for automatically decoding the challenges shown in MIOR CAPTCHAs. To see how this attack would work, I remind the reader that a video can be seen as a stream of single pictures that simply provides multiple views of a temporally evolving scene. It is well known that human observers perceive a naturally moving scene at a level of about thirty frames per second, and for this reason, video CAPTCHAs tend to use a comparable frame rate to provide a natural video experience that is not too jerky. Similarly, the challenge shown in the CAPTCHA is rendered in multiple frames to allow users to perceive and decode the codewords in an effortless manner. In the NuCaptcha scheme, for example, a single frame may contain the full codeword.

3.2.1 A Naïve Attack

Given this observation, one way to attack such schemes is to simply apply traditional OCR-based techniques that work well at defeating CR-still CAPTCHAs (e.g., (Mori and Malik, 2003;

Yan and Ahmad, 2007)). More specifically, choose k frames at random, and identify the foreground pixels of the codeword by comparing their color with a given reference color; notice the attacker would likely know this value since the users are asked to, for example, “type the RED moving characters”. Next, the length of the codeword can be inferred by finding the leftmost and rightmost pixels on the foreground. This in essence defines a line spanning over the foreground pixels (see Figure 3.2). The positions of the characters along the line can be determined by dividing the line into n equidistant segments, where n denotes the desired number of characters in the codeword. For each of the segments, compute the center of gravity of the foreground pixels in the vertical area of the image belonging to the segment. Lastly, select an image patch (of the expected size of the characters) around the centers of gravity of the segments, and feed each patch to a classifier. In my work, I use a neural network approach (Simard et al., 2003) because it is known to perform well at this object identification task. The neural network is trained in a manner similar to what I discuss in §3.2.3.

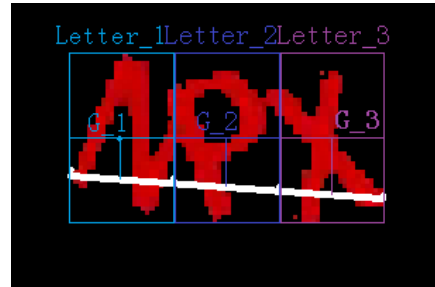


Figure 3.2: Naïve attack: Based on the foreground pixels, I find the longest horizontal distance (white line) and the mean value of vertical area (the respective bounding boxes above).

The above process yields a guess for each of the characters of the codeword in the chosen frames of the video. Let i denote the number of possible answers for each character. By transforming the score from the neural network into the probability p_{ijk} where the j -th character of the codeword corresponds to the i -th character in the k -th frame, I calculate the probability P_{ij} for each character $j = 1, \dots, n$ of the codeword over all k frames as $P_{ij} = \frac{1}{s_p} \sum_k p_{ijk}$ with $s_p = \sum_{i,j,k} p_{ijk}$. The choice that has the highest probability is selected as the corresponding character. With $k = 10$, this naïve attack resulted in a success rate of approximately 36% accuracy in correctly deducing

all three characters in the codewords of 4000 CAPTCHAs from NuCaptcha. While this relatively simple attack already raises doubts about the robustness of this new MIOR CAPTCHA, I now present a significantly improved attack that makes fewer assumptions about pixel invariants (Yan and El Ahmad, 2011) in the videos.

3.2.2 Exploiting Temporal Information

A clear limitation of the naïve attack is the fact that it is not easily generalizable and it is not robust to slight changes in the videos. In what follows, we make no assumption about a priori knowledge of the color of the codewords, nor do I assume that the centers of gravity for each patch are equidistant. To do so, I apply a robust segmentation method that utilizes temporal information to improve my ability to recognize the characters in the video.

A basic overview of my attack is shown in Figure 3.3. Given a MIOR CAPTCHA I extract the motion contained in the video using the concept of salient features. Salient features are characteristic areas of an image that can be reliably detected in several frames. To infer the motion of the salient feature points, I apply object tracking techniques (stage ❶). With a set of salient features at hand, I then use these features to estimate the color statistics of the background. Specifically, I use a Gaussian mixture model (Friedman and Russell, 1976), which represents the color statistics of the background through a limited set of Gaussian distributions. I use the color model of the background to measure, for all pixels in each frame, their likelihood of belonging to the background. Pixels with low likelihoods are then extracted as foreground pixels (stage ❷). The trajectories of the foreground pixels are then refined using information inferred about the color of these pixels, and a foreground color model is built. Next, to account for the fact that all characters of the codewords move independently, I segment the foreground into n segments as in the naïve attack (stage ❸). I select each image patch containing a candidate character and evaluate the patch using a neural network based classifier (Simard et al., 2003) (stage ❹). The classifier outputs a likelihood score that the patch contains a character. As a final enhancement, I incorporate a feedback mechanism in which I use high confidence inferences to improve low confidence detections of other patches.

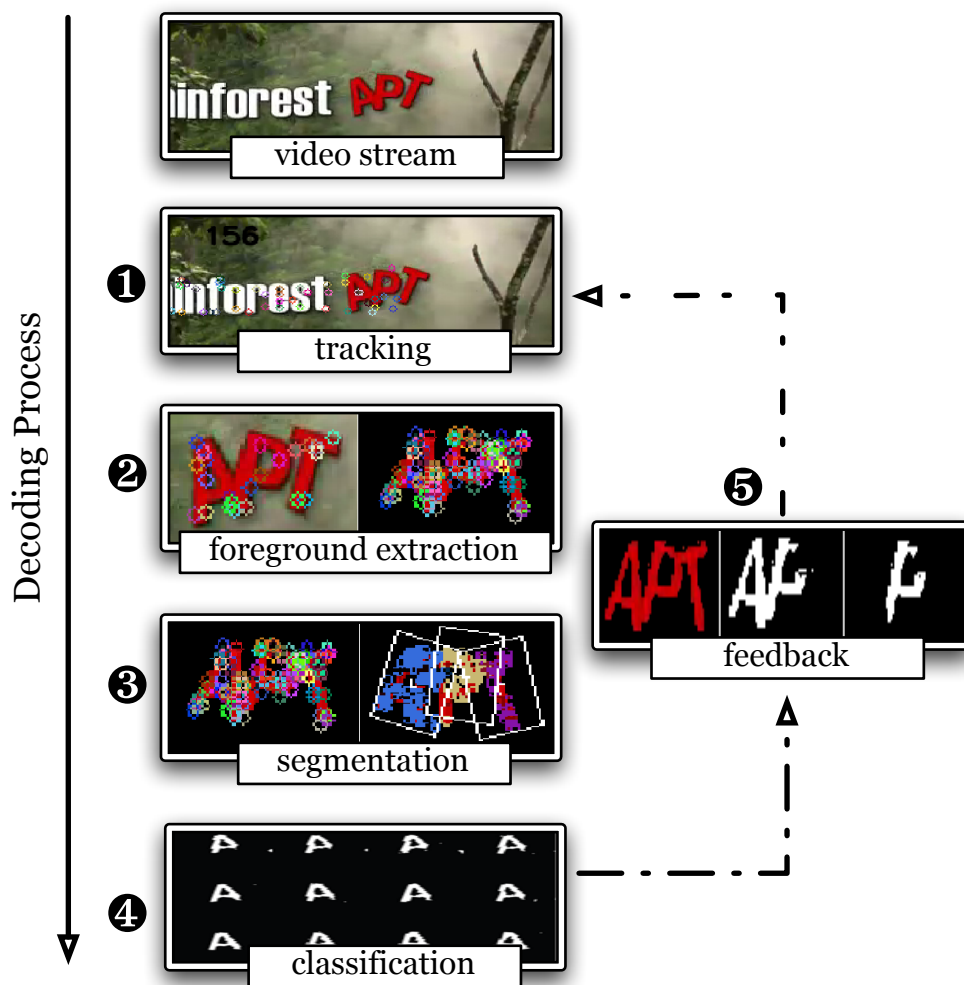


Figure 3.3: High-level overview of my attack. (This, and other figures, are best viewed in color.)

The net effect is that I reduce the distractions caused by mutually overlapping characters. Once all segments have been classified, I output my guess for all characters of the codeword. I now discuss the stages of my approach in more detail.

Detecting Salient Features and Their Motion (Stage ❶)

A well-known class of salient features in the computer vision community is gray value corners in images. In this chapter, I use the Harris corner detector (Harris and Stephens, 1988) for computing salient features, which uses the image gradient to identify points in the image with two orthogonal gradients of significant magnitude. An example of the detected corners is shown in Figure 3.4.



Figure 3.4: The circles depict salient features. These salient features are usually corners of an object or texture areas.

After identifying salient features in one frame of the video I now need to identify their respective position in the subsequent frames of the video. In general, there are two choices for identifying the corresponding salient features in the subsequent frames of the video. The first choice is to independently detect salient features in all frames and then compare them by using their image neighborhoods (patches) to identify correlating patches through an image based correlation (commonly called matching). The second class of methods leverages the small motion occurring in between two frames for an iterative search (commonly called tracking).

I opt for a tracking method given that tracking results for video are superior in accuracy and precision to matching results. Specifically, I deploy the well known KLT-tracking method (Lucas and Kanade, 1981), which is based on the assumption that the image of a scene object has a constant appearance in the different frames capturing the object (brightness constancy). The MIOR CAPTCHAs by NuCaptcha use constant colors on the characters of the codewords. This implies that the NuCaptcha frames are well suited for my method. Note that no assumption about the specific color is made; only constant appearance of each of the salient features is assumed. I return to this assumption later in Section 3.3.2.

Motion Trajectory Clustering (Stage ②)

In a typical video, the detected salient features will be spread throughout the image. In the case of NuCaptcha, the detected features are either on the background, the plain (i.e., non-codeword)

characters or the codeword characters. I am foremost interested in obtaining the information of the codeword characters. To identify the codeword characters I use their distinctive motion patterns as their motion is the most irregular motion in the video CAPTCHA. In the case of NuCaptcha, I take advantage of the fact that the motion trajectories of the background are significantly less stable (i.e., across consecutive frames) than the trajectories of the features on the characters. Hence I can identify background features by finding motion trajectories covering only a fraction of the sequence; specifically I assume presence for less than $l = 20$ frames. In my analysis, I observed little sensitivity with respect to l .

Additionally, given that all characters (plain and codeword) move along a common trajectory, I can further identify this common component by linearly fitting a trajectory to their path. Note that the centers of the rotating codeword characters still move along this trajectory. Accordingly, I use the distinctive rotation of the codeword characters to identify any of their associated patterns by simply searching for the trajectories with the largest deviation from the more common motion trajectory. This identifies the pixels belonging to the codeword characters as well as the plain characters. Additionally, the features on the identified codeword characters allow us to obtain the specific color of the codeword characters without knowing the color a priori (see Figure 3.5).

Knowing the position of the codeword characters allows us to learn a foreground color model. I use a Gaussian mixture model for the foreground learning, which in my case has a single moment corresponding to the foreground color.¹ Additionally, given the above identified salient features on the background, I also learn a Gaussian mixture for the background, thereby further separating the characters from the background.

At this point, I have isolated the trajectories of codeword characters, and separated the codewords from the background (see Figure 3.6). However, to decide which salient features on the codeword characters belong together, I required additional trajectories. To acquire these, I simply relax the constraint on the sharpness of corners I care about (i.e., I lower the threshold for the Harris corner detection algorithm) and rerun the KLT-tracking on the new salient features. This yields

¹ In the case where the foreground characters have varying appearance, I simply use multiple modes.



Figure 3.5: (Top): Initial optical flow. (Middle): salient points with short trajectories in background are discarded. (Lower): Trajectories on non-codeword characters are also discarded.

significantly more trajectories for use by my segmentation algorithm. Notice how dense the salient features are in Figure 3.7. Note also that since the foreground extraction step provides patches that are not related to the background, I can automatically generate training samples for my classifier, irrespective of the various backgrounds the characters are contained in.

Segmentation (Stage ③)

To segment the derived trajectories into groups, I use k -means clustering (Jain et al., 1999). I chose this approach over other considerations (e.g., mean-shift (Ray and Turi, 1999) based



Figure 3.6: Example foreground extraction.

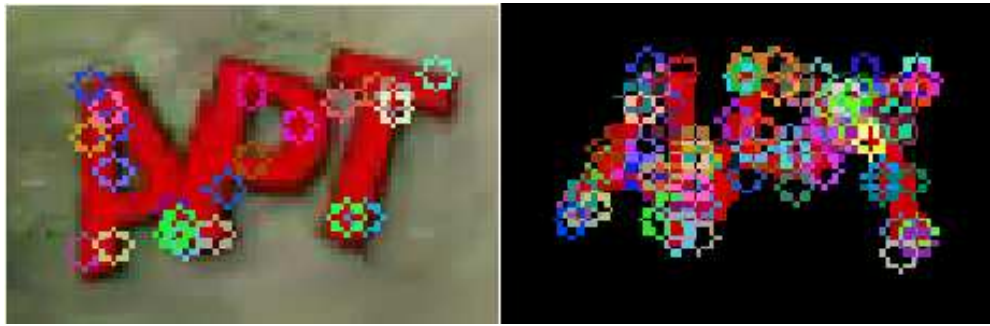


Figure 3.7: re-running tracking with a lower threshold on corner quality: Left: before modification. Right: after modification.

clustering, or RANSAC (Fischler and Bolles, 1981a) based clustering (Yan and Pollefeys, 2007)) because of its simplicity, coupled with the fact that I can take advantage of my knowledge of the desired number of characters (i.e., k), and use that to help guide the clustering procedure. I cannot, however, apply the standard k -means approach directly since it relies on Euclidean distances, where each sample is a point. In my case, I need to take the relationship between frames of the video sequence into consideration, and so I must instead use each trajectory as an observation. That is, I cluster the different trajectories. However, this results in a non-Euclidean space because different trajectories have different beginning and ending frames. To address this problem, I utilize the rigidity assumption (Ullman, 1983) and define a distance metric for trajectories that takes into consideration their spatial distance, as well as the variation of their spatial distance. The result is a robust technique that typically converges within 5 iterations when $k = 3$, and 20 iterations (on average) when $k = 23$. A sample output of this stage is shown in Figure 3.8.

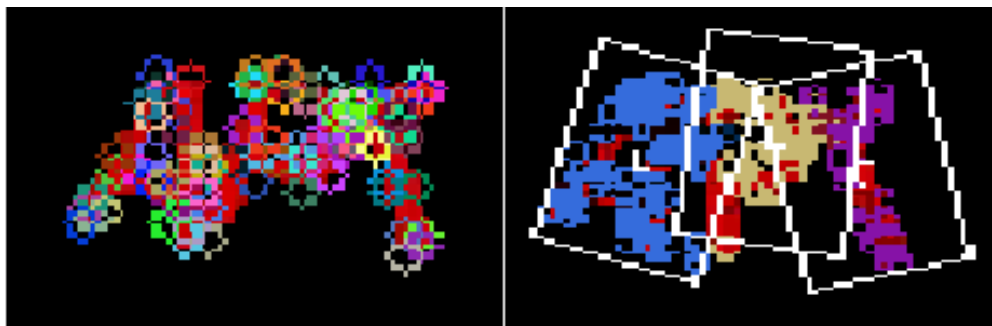


Figure 3.8: Left: before segmentation. Right: trajectories are marked with different colors and bounding boxes are calculated based on the center of the trajectories and the orientation of the points. The red points denote areas with no trajectories.

3.2.3 Codeword Extraction and Classification

Given the center and orientation of each codeword character, the goal is to figure out exactly what that character is. For this task, I extract a fixed-sized area around each character (as in Figure 3.8), and supply that to my classification stage. Before doing so, however, I refine the patches by deleting pixels that are too close to the trajectories of adjacent characters.

As mentioned earlier, I use a neural network for classifying the refined patches. A neural network is a mathematical model or computational model that is inspired by the structure of a biological neural network. The training of a neural network is based on the notion of the possibility of learning. Given a specific task to solve, and a class of functions, learning in this context means using a set of observations to find a function which solves the task in some optimal sense.

Optimization: While the process outlined in stages ❶-❹ works surprisingly well, there are several opportunities for improvement. Perhaps one of the most natural extensions is to utilize a feedback mechanism to boost recognition accuracy. The idea I pursue is based on the observation that an adversary can leverage her confidence about what particular patches represent to improve her overall ability to break the CAPTCHA. Specifically, I find and block the character that I am most confident about. The basic idea is that although I may not be able to infer all the characters at once, it is very likely that I can infer some of the characters. By masking the character that I am most confident

about, I can simplify the problem into one of decoding a codeword with fewer characters; which is easier to segment and recognize.



Figure 3.9: Iterative decoding of a CAPTCHA.

The most confident character can be found using the probability score provided by the classifier, although it is non-trivial to do so without masking out too much of the other characters. I solve this problem as follows. In order to block a character, I try to match it with templates of each character that can be gained by learning. One way to do that is to match scale-invariant feature transforms (SIFT) between the patch and a reference template. While SIFT features can deal with scaling, rotation and translation of characters, there are times when some frames have insufficient SIFT features. My solution is to find a frame with enough features to apply SIFT, and then warp the template to mask the target character in that frame. Once found, this frame is used as the initial position in an incremental alignment approach based on KLT tracking. Essentially, I combine the benefits of SIFT and KLT to provide a video sequence where the character I am most confident about is omitted. At that point, I rerun my attack, but with one fewer character. This process is repeated until I have no characters left to decode. This process is illustrated in Figure 3.9.

Runtime: Our implementation is based on a collection of modules written in a mix of C++ and Matlab code. I make extensive use of the Open Source Computer Vision library (OpenCV). My un-optimized code takes approximately 30s to decode the three characters in a MIOR CAPTCHA when the feedback loop optimization (in stage ④) is disabled. With feedback enabled, processing time increases to 250s. The bottleneck is in the incremental alignment procedure (written in Matlab).

3.3 Evaluation

I now discuss the results of experiments I performed on MIOR CAPTCHAs. Specifically, the first set of experiments are based on video sequences downloaded off the demo page of NuCaptcha’s website. On each visit to the demo page, a CAPTCHA with a random 3-character codeword is displayed for 6 seconds before the video loops. The displayed CAPTCHAs were saved locally using a Firefox plugin called NetVideoHunter. I downloaded 4500 CAPTCHAs during November and December of 2011.

Attack Strategy	Single Character Accuracy	3-Character Accuracy
Naïve	68.5% (8216/12000)	36.3% (1450/4000)
Enhanced (<i>no feedback</i>)	90.0% (540/600)	75.5% (151/200)
Enhanced (<i>with feedback</i>)	90.3% (542/600)	77.0% (154/200)

Table 3.1: Reconstruction accuracy for various attacks.

The collected videos contain CAPTCHAs with all 19 backgrounds in use by NuCaptcha as of December 2011. In each of these videos, the backgrounds are of moving scenes (e.g., waves on a beach, kids playing baseball, etc.) and the text in the foreground either moves across the field of view or in-place. I painstakingly labeled each of the videos by hand to obtain the ground truth. I note that while NuCaptcha provides an API for obtaining CAPTCHAs, I opted not to use that service as I did not want to interfere with their service in any way. In addition, my second set of experiments examine several countermeasures against my attacks, and so for ethical reasons, I opted to perform such experiments in a controlled manner rather than with any in-the-wild experimentation. These countermeasures are also evaluated in my user study (§3.4).

3.3.1 Results

The naïve attack was analyzed on 4000 CAPTCHAs. Due to time constraints, the extended attack (with and without the feedback optimization) were each analyzed on a random sample of 500 CAPTCHAs. To determine an appropriate training set size, I varied the number of videos as well as the number of extracted frames and examined the recognition rate. The results (not

shown) show that while accuracy steadily increased with more training videos (e.g., 50 versus 100 videos), I only observed marginal improvement when the number of training patches taken from each video exceeded 1500. In the subsequent analyses, I use 300 video sequences for training (i.e., 900 codeword characters) and for each detected character, I select 2 frames containing that character (yielding 1800 training patches in total). I use dense SIFT descriptors (Vedaldi and Fulkerson, 2010) as the features for each patch (i.e., a SIFT descriptor is extracted for each pixel in the patch, and concatenated to form a feature vector). The feature vectors are used to train the neural network. For testing, I choose a *different* set of 200 CAPTCHAs, almost evenly distributed among the 19 backgrounds. The accuracy of the attacks (in §3.2) are given in Table 3.1.

The result indicate that the robustness of these MIOR CAPTCHAs are far weaker than one would hope. In particular, my automated attacks can completely decode the CAPTCHAs *more than three quarters of the time*. In fact, my success rates are even higher than some of the OCR-based attacks on CR-still CAPTCHAs (Mori and Malik, 2003; Yan and Ahmad, 2007; Golle, 2008; Bursztein et al., 2011b). There are, however, some obvious countermeasures that designers of MIOR CAPTCHAs might employ.

3.3.2 Mitigation

To highlight some of the tensions that exists between the security and usability of MIOR CAPTCHAs, we explore a series of possible mitigations to my attacks. In order to do so, I generate video CAPTCHAs that closely mimic those from NuCaptcha. In particular, I built a framework for generating videos with characters that move across a background scene with constant velocity in the horizontal direction, and move up and down harmonically. Similar to NuCaptcha, the characters of the codeword also rotate. My framework is tunable, and all the parameters are set to the defaults calculated from the original videos from NuCaptcha (denoted *Standard*). I refer the interested reader to Appendix A for more details on how I set the parameters. Given this framework, I explore the following defenses:

- *Extended*: the codeword consists of $m > 3$ random characters moving across a dynamic scene.
- *Overlapping*: same as the *Standard* case (i.e., $m = 3$), except characters are more closely overlapped.
- *Semi-Transparent*: identical to the *Standard* case, except that the characters are semi-transparent.
- *Emerging objects*: a different MIOR CAPTCHA where the codewords are 3 characters but created using an “Emerging Images” (Mitra et al., 2009) concept (see below).



Figure 3.10: Extended case. Top: scrolling; bottom: in-place.

Increasing the number of random characters shown in the CAPTCHA would be a natural way to mitigate my attack. Hence, the *Extended* characters case is meant to investigate the point at which the success rate of my attacks fall below a predefined threshold. An example is shown in Figure 3.10. Similarly, I initially thought that increasing the overlap between consecutive characters (i.e., the *Overlapping* defense, Fig. 3.11) might be a viable alternative. I estimate the degree that two characters overlap by the ratio of the horizontal distance of their centers and their average width. That is, suppose that one character is 20 pixels wide, and the other is 30 pixels wide. If the horizontal distance of their centers is 20, then their overlap ratio is computed as $20 / \frac{20+30}{2} = 0.8$. The smaller this overlap ratio, the more the characters overlap. A ratio of 0.5 means that the middle

character is completely overlapped in the horizontal direction. In both the original CAPTCHAs from NuCaptcha and my *Standard* case, the overlap ratio is 0.95 for any two adjacent characters.



Figure 3.11: Overlapping characters (with ratio = 0.49).

The *Semi-Transparent* defense is an attempt to break the assumption that the foreground is of constant color. In this case, foreground extraction (stage ②) will be difficult. To mimic this defense strategy, I adjust the background-to-foreground pixel ratio. An example is shown in figure 3.12.



Figure 3.12: Semi-transparent: 80% background to 20% foreground pixel ratio. (Best viewed in color.)

The final countermeasure is based on the notion of *Emerging Images* proposed by Mitra et al. (2009). Emergence refers to “the unique human ability to aggregate information from seemingly meaningless pieces, and to perceive a whole that is meaningful” (Mitra et al., 2009).² The concept has been exploited in Computer Graphics to prevent automated tracking by computers, while simultaneously allowing for high recognition rates in humans because of my remarkable visual system. I apply the concepts outlined by Mitra et al. (Mitra et al., 2009) to generate CAPTCHAs that are resilient to my attacks. The key differences between my implementation and the original paper is that my input is 2D characters instead of 3D objects, and I do not have the luxury of incorporating shadow information. My Emerging CAPTCHAs are constructed as follows:

1. I build a noisy frame I_{bg} by creating an image with each pixel following a Gaussian distribution.

I blur the image such that the value of each pixel is related to nearby pixels. I also include

² Readers can view videos of the Emerging Images concept (Mitra et al., 2009) at http://graphics.stanford.edu/~niloy/research/emergence/emergence_image_siga_09.html.

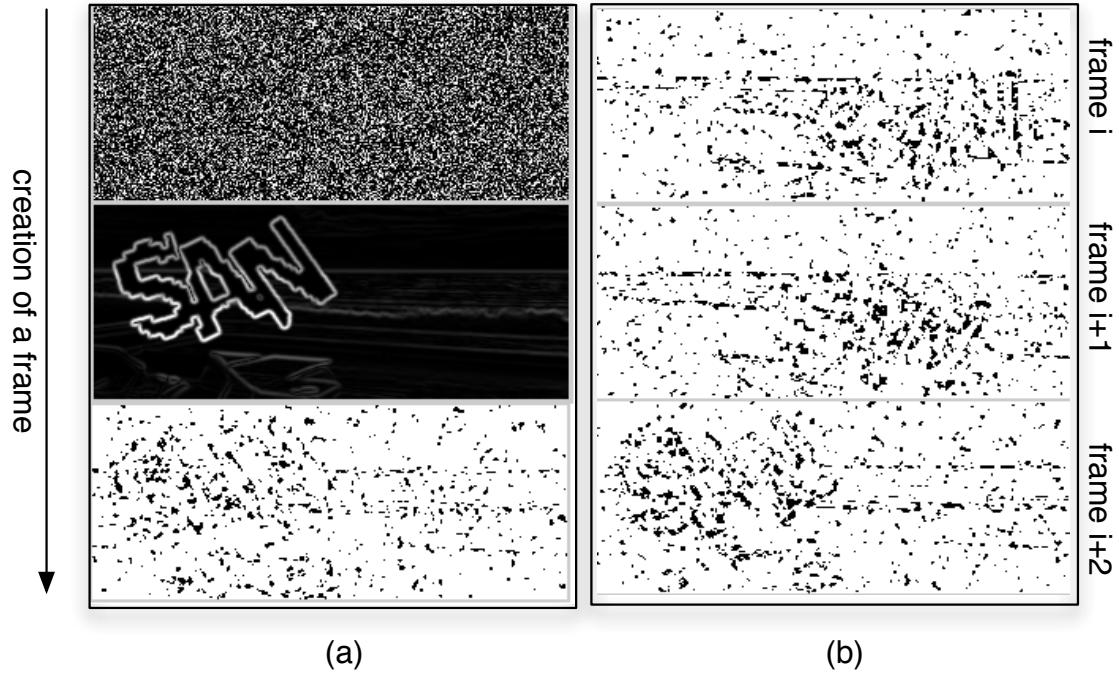


Figure 3.13: Emerging CAPTCHA. (a) Top: noisy background frame. Middle: derivative of foreground image. Bottom: single frame for an Emerging CAPTCHA. (b) Successive frames.

time correspondence by filtering in the time domain. That is, each frame is a mixture of a new noisy image and the last frame.

2. I generate an image I_{fg} similar to that in NuCaptcha. I then find the edges in the image by calculating the norm of derivatives of the image.
3. I combine I_{bg} and I_{fg} by creating a new image I where each pixel in I is defined as $I(x, y) := I_{bg}(x, y) * \exp(\frac{I_{fg}}{const})$, where $\exp(x)$ is the exponential function. In this way, the pixels near the boundary of characters in I are made more noisy than other pixels.
4. I define a constant threshold $t < 0$. All pixel values in I that are larger than t are made white. All the other pixels in I are made black.

The above procedure results in a series of frames where no single frame contains the codeword in a way that is easy to segment. The pixels near the boundaries of the characters are also more likely to be blacker than other pixels, which the human visual system somehow uses to identify the structure from motion. This feat remains challenging for computers since the points near the

boundaries change color randomly, making it difficult, if not impossible, to track, using existing techniques. An illustration is shown in Figure 3.13. To the best of my knowledge, I provide the first concrete instantiation of the notion of Emerging Images applied to CAPTCHAs, as well as a corresponding lab-based usability study (§3.4).

I refer interested readers to <http://www.cs.unc.edu/videocaptcha/> for examples of the mitigation strategies I explored.

3.3.2.1 Results

I now report on the results of running attacks on CAPTCHAs employing the aforementioned defenses. Figure 3.14 depicts the results for the *Extended* defense strategy. In these experiments, I generated 100 random CAPTCHAs for each $m \in [3, 23]$. My results clearly show that simply increasing the codeword length is not necessarily a viable defense. In fact, even at 23 characters, my success rate is still 5%, on average.

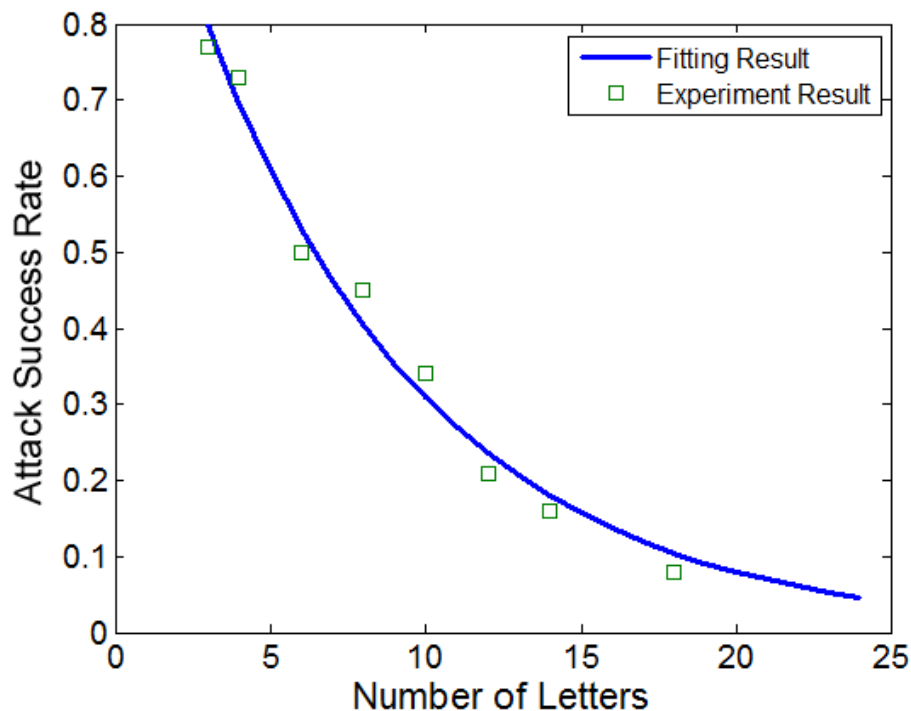


Figure 3.14: Attack success as a function of codeword length.

Figure 3.15 shows the results for the *Overlapping* defense strategy. As before, the results are averaged over 100 sequences per point. The graph shows that the success rate drops steadily as the overlap ratio decreases (denoted as “sensitivity” level in that plot). Interestingly, NuCaptcha mentions that this defense strategy is in fact one of the security features enabled by its behavioral analysis engine. The images provided on their website for the “*very secure*” mode, however, have an overlap ratio of 0.78, which my attacks would still be able to break more than 50% of the time.³ My success rate is still relatively high (at 5%) even when the overlap ratio is as low as 0.49. Recall that, at that point, the middle character is 100% overlapped, and others are 51% overlapped.

Figure 3.15 also shows the results for the *Semi-Transparent* experiment. In that case, I varied the transparency of the foreground pixel from 100% down to 20%. Even when the codewords are barely visible (to the human eye), I am still able to break the CAPTCHAs 5% of the time. An example of one such CAPTCHA (with a background to foreground ratio of 80 to 20 percent) was shown earlier in Figure 3.12.

Lastly, I generated 100 CAPTCHAs based on my implementation of the Emerging Images concept. It comes as no surprise that the attacks in this chapter were not able to decode a single one of these challenges — precisely because these CAPTCHAs were specifically designed to make optical flow tracking and object segmentation difficult. From a security perspective, these MIOR CAPTCHAs are more robust than the other defenses I examined. I return to that discussion in §3.5.

3.3.2.2 Discussion

The question remains, however, whether for any of the defenses, parameters could be tuned to increase the robustness and still retain *usability*. I explore precisely that question next. That said, the forthcoming analysis raises interesting questions, especially as it relates to the robustness of CAPTCHAs. In particular, there is presently no consensus on the required adversarial effort a CAPTCHA should present, or the security threshold in terms of success rate that adversaries should be held below. For example, Chellapilla et al. (Chellapilla et al., 2005b) state: “automated attacks

³ See the Security Features discussed at <http://www.nucaptcha.com/features/security-features>, 2012.

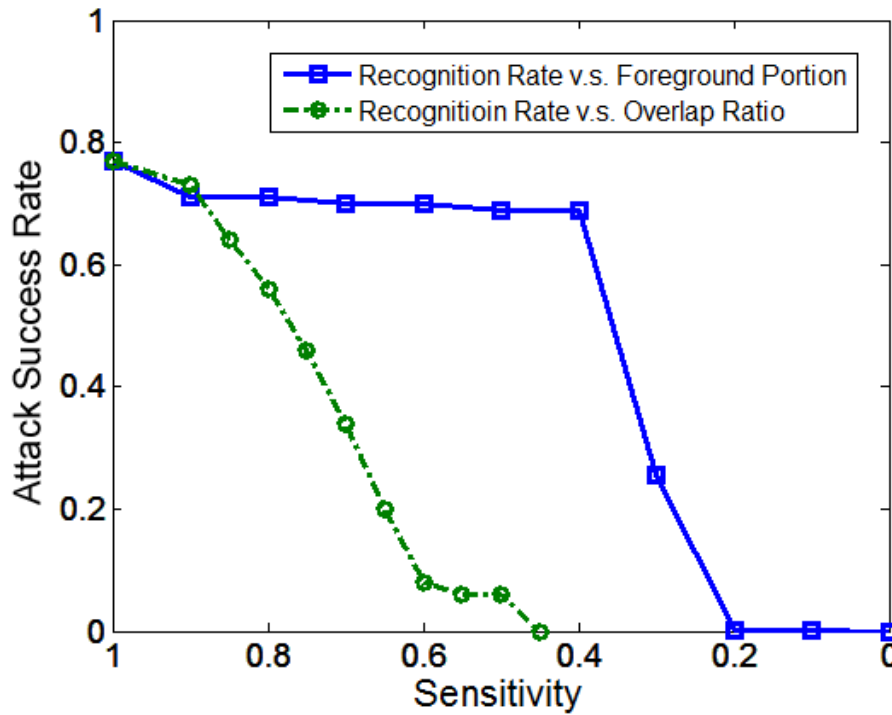


Figure 3.15: Attack success rate against *Overlapping* and *Semi-Transparent* defenses. Sensitivity refers to the overlap ratio (circles) or the background-to-foreground ratio (squares).

should not be more than 0.01% successful but the human success rate should be at least 90%.” Others argue that “if it is at least as expensive for an attacker to break the challenge by machine than it would be to pay a human to take the CAPTCHA, the test can be considered secure” (Hidalgo and Alvarez, 2011). Zhu et al. (Zhu et al., 2010) use the metric that the bot success rate should not exceed 0.6%.

In the course of my pilot studies, it became clear that if the parameters for the *Extended*, *Overlapping*, and *Semi-Transparent* countermeasures are set too stringently (e.g., to defeat automated attacks 99% of the time), then the resulting MIOR CAPTCHAs would be exceedingly difficult for humans to solve. Therefore, to better measure the tension between usability and security, I set the parameters for the videos (in §3.4) to values where my attacks have a 5% success rate, despite that being intolerably high for practical security. Any CAPTCHA at this parametrization, which is found to be unusable, is thus entirely unviable.

3.4 User study

We now report on an IRB-approved user study with 25 participants that we conducted to assess the usability of the aforementioned countermeasures. If the challenges produced by the countermeasures prove too difficult for both computers and humans to solve, then they are not viable as CAPTCHA challenges. We chose a controlled lab study because besides collecting quantitative performance data, it gave us the opportunity to collect participants’ impromptu reactions and comments, and allowed us to interview participants about their experience. This type of information is invaluable in learning *why* certain mitigation strategies are unacceptable or difficult for users and learning which strategies are deemed most acceptable. Additionally, while web-based or Mechanical Turk studies may have allowed us to collect data from more participants, such approaches lack the richness of data available when the experimenter has the opportunity to interact with the participants one-on-one. Mechanical Turk studies have previously been used in CAPTCHA research (Bursztein et al., 2010) when the goal of the studies are entirely performance-based. However, since we are studying new mitigation strategies, I felt that it was important to gather both qualitative and quantitative data for a more holistic perspective.



Figure 3.16: Three backgrounds used for the challenges, shown for the *Semi-Transparent* variant.

3.4.1 Methodology

I compared the defenses in §3.3.2 to a *Standard* approach which mimics NuCaptcha’s design. In these CAPTCHAs the video contains scrolling text with 2-3 words in white font, followed by three random red characters that move along the same trajectory as the white words. Similar to NuCaptcha, the red characters (i.e., the codewords) also independently rotate as they move. For the

Extended strategy, I set $m = 23$. All 23 characters are continuously visible on the screen. During pilot testing, I also tried a scrolling 23-character variation of the *Extended* scheme. However, this proved extremely difficult for users to solve and they voiced strong dislike (and outrage) for the variation. For the *Overlapping* strategy, I set the ratio to be 0.49. Recall that at this ratio, the middle character is overlapped 100% of the time, and the others are 51% overlapped. For the *Semi-Transparent* strategy, I set the ratio to be 80% background and 20% foreground. For all experiments, I use the same alphabet (of 20 characters) in NuCaptcha's original videos.

A *challenge* refers to a single CAPTCHA puzzle to be solved by the user. Each challenge was displayed on a 6-second video clip that used a canvas of size 300×126 and looped continuously. This is the same specification used in NuCaptcha's videos. Three different HD video backgrounds (of a forest, a beach, and a sky) were used. Some examples are shown in Figure 3.16. Sixty challenges were generated for each variation (20 for each background, as applicable).

I also tested the *Emerging* strategy. The three-character codeword was represented by black and white pixel-based noise as described in §3.3.2. Sixty challenges were generated using the same video parameters as the other conditions.

The twenty-five participants were undergraduate, graduate students, staff and faculty (15 males, 10 females, mean age 26) from a variety of disciplines. A within-subjects experimental design was used, where each participant had a chance to complete a set of 10 CAPTCHAs for each strategy. The order of presentation for the variations was counterbalanced according to a 5×5 Latin Square to eliminate biases from learning effects; Latin Squares are preferred over random ordering of conditions because randomization could lead to a situation where one condition is favored (e.g., appearing in the last position more frequently than other conditions, giving participants more chance to practice). Within each variation, challenges were randomly selected.

A simple web-based user interface was designed where users could enter their response in the textbox and press submit, could request a new challenge, or could access the help file. Indication of correctness was provided when users submitted their responses, and users were randomly shown the next challenge in the set. Immediately after completing the 10 challenges for a variation, users

were asked to complete a paper-based questionnaire collecting their perception and opinion of that variation. At the end of the session, a brief interview was conducted to gather any overall comments. Each participant completed their session one-on-one with the experimenter. A session lasted at most 45 minutes and users were compensated \$15 for their time.

3.4.2 Data Collection

The user interface was instrumented to log each user's interactions with the system. For each challenge, the user's textual response, the timing information, and the outcome was recorded. A challenge could result in three possible outcomes: success, error, or skipped. Questionnaire and interview data was also collected.

3.4.3 Analysis

My analysis focused on the effects of five different CAPTCHA variants on outcomes and solving times. I also analyzed and reviewed questionnaire data representing participant perceptions of the five variants. I used several statistical tests and the within-subjects design of my study impacted my choice of statistical tests; in each case the chosen test accounted for the fact that I had multiple data points from each participant. In all of my tests, I chose $p < 0.05$ as the threshold for determining statistical significance.

One-way repeated-measures ANOVAs (Lazar et al., 2010) were used to evaluate aggregate differences between the means for success rates and times. When the ANOVA revealed a significant difference, I used post-hoc Tukey HSD tests (Lowry, 1998) to determine between which pairs the differences occurred. Here, I was interested only in whether the four proposed mitigation strategies differed from the *Standard* variant, so I report only on these four cases.

My questionnaires used Likert-scale responses to assess agreement with particular statements (1 - Strongly Disagree, 10 - Strongly Agree). To compare this ordinal data, I used the non-parametric Friedman's Test (Lowry, 1998). When overall significant differences were found, I used post-hoc

Pairwise Wilcoxon tests with Bonferroni correction to see which of the four proposed variants differed from the *Standard* variant.

Outcomes: Participants were presented with 10 challenges of each variant. Figure 3.17 shows a stacked bar graph representing the mean number of success, error, and skipped outcomes. To be identified as a *Success*, the user’s response had to be entirely correct. An *Error* occurred when the user’s response did not match the challenge’s solution. A *Skipped* outcome occurred when the participant pressed the “Get A New Challenge” button and was presented with a different challenge. I observe differences in the outcomes, with the *Standard* variant being most successful and the *Semi-Transparent* variant resulting in the most skipped outcomes.

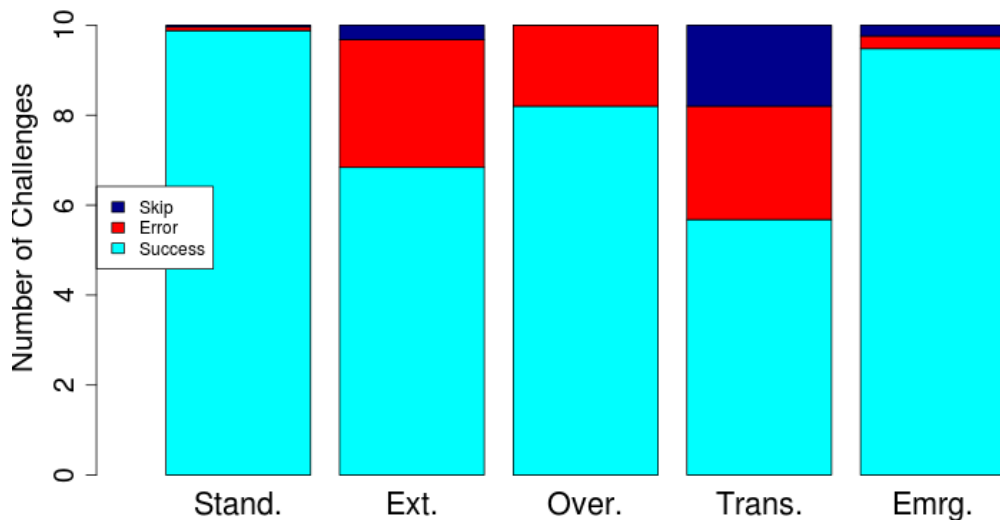


Figure 3.17: Mean number of success, error, and skipped outcomes for *Standard*, *Extended*, *Overlapping*, *Semi-Transparent* and *Emerging* variants, respectively.

For the purposes of my statistical tests, errors and skipped outcomes were grouped since in both cases the user was unable to solve the challenge. Each participant was given a score comprising the number of successful outcomes for each variant (out of 10 challenges).⁴

A one-way repeated-measure ANOVA showed significant differences between the five variants ($F(4, 120) = 29.12, p < 0.001$). I used post-hoc Tukey HSD tests to see whether any of the differences occurred between the *Standard* variant and any of the other four variants. The tests

⁴ One participant opted to view only six challenges in each of the *Extended* and *Emerging* variants. I count the remaining four as skips.

showed a statistically significant difference between all pairs except for the *Standard*↔*Emerging* pair. This means that the *Extended*, *Overlapping*, and *Semi-Transparent* variants had a significantly lower number of successes than the *Standard* variant, while *Emerging* variant showed no difference.

Time to Solve: The time to solve was measured as the time between when the challenge was displayed to when the response was received. This included the time to type the answer (correctly or incorrectly), as well as the time it took the system to receive the reply (since the challenges were served from my local server, transmission time was negligible). Times for skipped challenges were not included since users made “skip” decisions very quickly and this may unfairly skew the results towards shorter mean times. I include challenges that resulted in errors because in these cases participants actively tried to solve the challenge. The time distributions are depicted in Figure 3.18 using boxplots. Notice that the *Extended* variant took considerably longer to solve than the others.

I examined the differences in mean times using a one-way repeated-measure ANOVA. The ANOVA showed overall significant differences between the five variants ($F(4, 120) = 112.95, p < 0.001$). Once again, I compared the *Standard* variant to the others in my post-hoc tests. Tukey HSD tests showed no significant differences between the *Standard*↔*Emerging* or *Standard*↔*Overlapping* pairs. However, significant differences were found for the *Standard*↔*Semi-Transparent* and *Standard*↔*Extended* pairs. This means that the *Semi-Transparent* and *Extended* variants took significantly longer to solve than the *Standard* variant, but the others showed no differences.

Skipped outcomes: The choice of background appears to have especially impacted the usability of the *Semi-Transparent* variant. Participants most frequently skipped challenges for the *Semi-Transparent* variant and found the Forest background especially difficult to use. Many users would immediately skip any challenge that appeared with the Forest background because the transparent letters were simply too difficult to see. For the *Semi-Transparent* variant, 35% of challenges presented on the Forest background were skipped, compared 17-18% of challenges using the other two backgrounds. Participants’ verbal and written comments confirm that they found the Forest background very difficult, with some users mentioning that they could not even find the letters as they scrolled over some parts of the image.

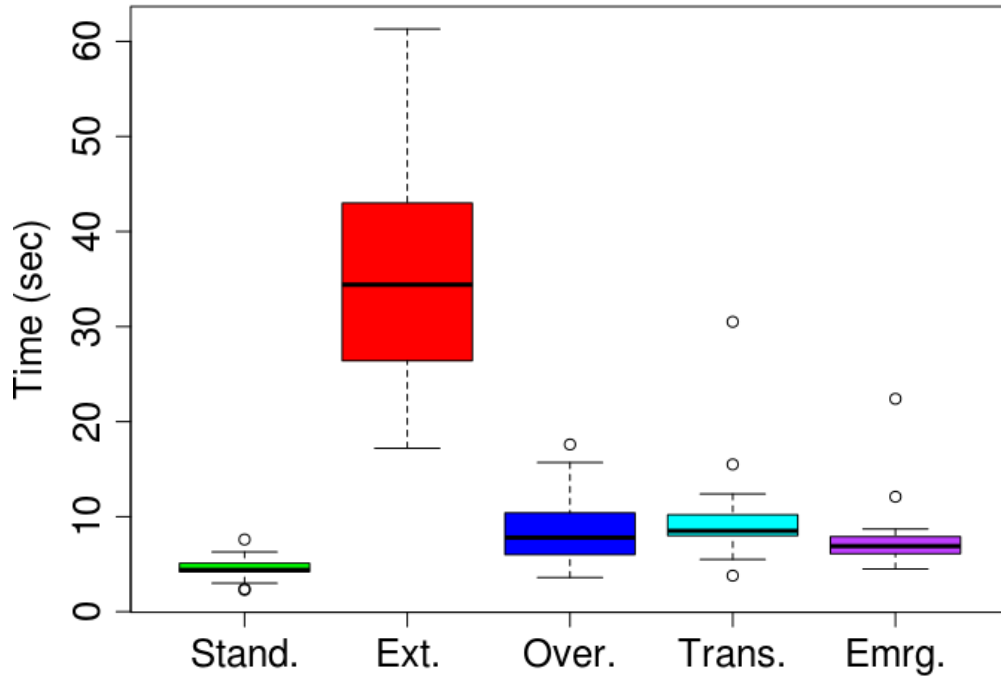


Figure 3.18: Time taken to solve the MBOR CAPTCHAs.

Errors: Figure 3.19 shows the distribution of errors. It shows that the majority of errors were made on the middle characters of the challenge. I also examined the types of errors, and found that most were mistakes between characters that have similar appearances. The most commonly confused pairs were: S/5, P/R, E/F, V/N, C/G, and 7/T. About half of the errors for the *Extended* variant were due to confusing pairs of characters, while the other half involved either missing letters or including extra ones. For the other variants, nearly all errors were due to confusing pairs of characters.

User perception: Immediately after completing the set of challenges for each variant, participants completed a Likert-scale questionnaire to collect their opinion and perception of that variant. For each variant, participants were asked to rate their agreement with the following statements:

1. It was easy to accurately solve the challenge
2. The challenges were easy to understand
3. This CAPTCHA mechanism was pleasant to use

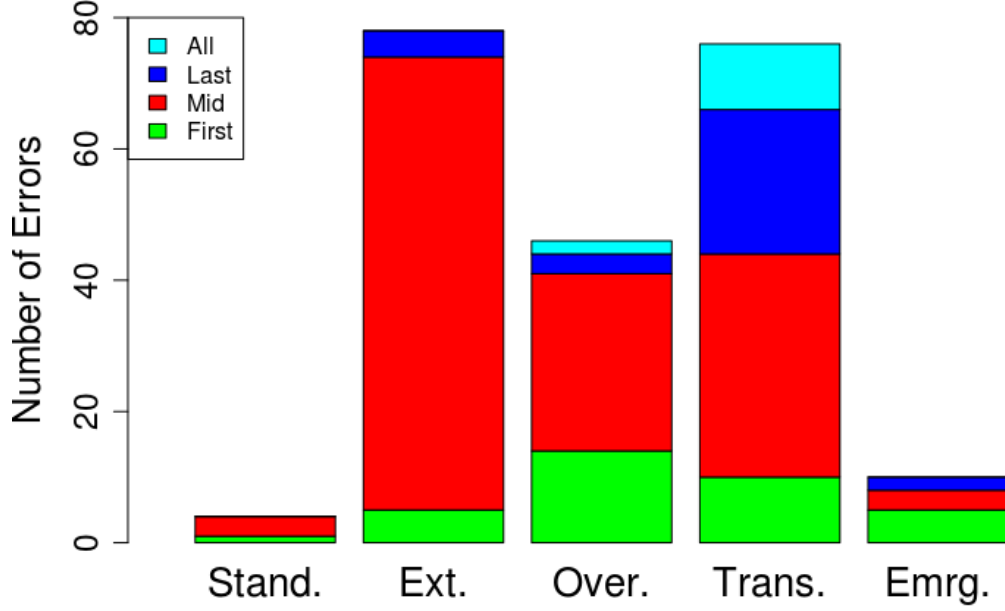


Figure 3.19: Location of errors within the codewords.

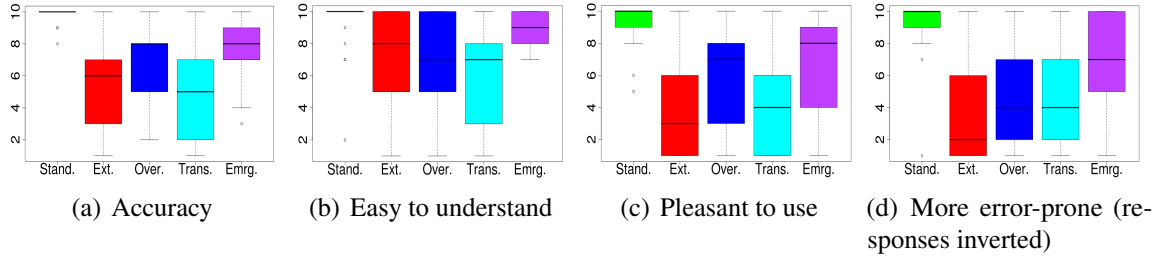


Figure 3.20: Likert-scale responses: 1 is most negative, 10 is most positive.

4. This CAPTCHA mechanism is more prone to mistakes than traditional text-based CAPTCHAs

Figure 3.20 shows boxplots representing users' responses. Since Q.4 was negatively worded, responses were inverted for easier comparisons. In all cases, higher values on the y-axis indicate a more favorable response.

The results show that users clearly preferred the *Standard* variant and rated the others considerably lower on all subjective measures. Friedman's Tests showed overall significant differences for each question ($p < 0.001$). Pairwise Wilcoxon Tests with Bonferroni correction were used to assess differences between the *Standard* variant and each of the other variants. Significant differences were found between each pair compared. The only exceptions are that users felt that the *Extended*

and *Emerging* variants were no more difficult to understand (Question 2) than the *Standard* variant. This result appears to contradict the results observed in Figure 3.20 and I believe that this is because the Wilcoxon test compares ranks rather than means or medians.

Comments: Participants had the opportunity to provide free-form comments about each variant and offer verbal comments to the experimenter. Samples are included in Appendix B. Participants clearly preferred the *Standard* variant, and most disliked the *Extended* variant. Of the remaining schemes, the *Emerging* variant seemed most acceptable although it also had its share of negative reactions (e.g., one subject found it to be hideous).

3.5 Summary and Concluding Remarks

My attack inherently leverages the temporal information in moving-image object recognition (MIOR) CAPTCHAs, and also exploits the fact that only object recognition of known objects is needed. My methods also rely on a reasonably consistent appearance or slowly varying appearance over time. That said, they can be applied to any set of known objects or narrowly defined objects under affine transformations that are known to work well with detection methods in computer vision (Viola and Jones, 2001). For the specific case of NuCaptcha, I showed that not only are there inherent weaknesses in the current MIOR CAPTCHA design, but that several obvious countermeasures (e.g., extending the length of the codeword) are not viable attack countermeasures. More importantly, my work highlights the fact that the choice of underlying hard problem by NuCaptcha’s designers was misguided; its particular implementation falls into a solvable subclass of computer vision problems.

In the case of emergent CAPTCHAs, my attacks fail for two main reasons. First, in each frame there are not enough visual cues that help distinguish the characters from the background. Second, the codewords have no temporally consistent appearance. Combined, these two facts pose significant challenges to existing computer vision methods, which typically assume reasonably consistent appearance and visually distinctive foregrounds (Yilmaz et al., 2006). Nevertheless, my

user study showed that people had little trouble solving these CAPTCHAs. This bodes well for emergent CAPTCHAs—for today’s attacks.

Looking towards the future, greater robustness would result if MIOR CAPTCHAs required automated attacks to perform classification, categorization of classes with large inner class variance, or to identify higher level semantics to understand the presented challenge. Consider, for example, the case where the user is presented with two objects (a person and a truck) at the same scale, and asked to identify which one is larger. To succeed, the automated attack would need to determine the objects (without prior knowledge of what the objects are of) and then understand the relationship. Humans can perform this task because of the inherent priors learned in daily life, but this feat remains a daunting problem in computer vision. Therefore, this combination seems to offer the right balance and underscores the ideas put forth by Naor (Naor, 1996) and von Ahn et al. (Ahn et al., 2003)—i.e., it is prudent to employ hard (and useful) underlying AI problems in CAPTCHAs since it leads to a win-win situation: either the CAPTCHA is not broken and there is a way to distinguish between humans and computers, or it is broken and a useful problem is solved.

CHAPTER 4: DEFEATING FACE LIVENESS DETECTION WITH VIRTUAL MODELS

4.1 Introduction

Over the past few years, face authentication systems have become increasingly popular as an enhanced security feature in both mobile devices and desktop computers. As the underlying computer vision algorithms have matured, many application designers and nascent specialist vendors have jumped in and started to offer solutions for mobile devices with varying degrees of security and usability. Other more well-known players, like Apple and Google, are posed to enter the market with their own solutions, having already acquired several facial recognition software companies¹. While the market is segmented based on the type of technology offered (*e.g.*, 2D facial recognition, 3D recognition, and facial analytics/face biometric authentication), Gartner research estimates that the overall market will grow to over \$6.5 billion in 2018 (compared to roughly \$2 billion today) (Gartner, 2014).

With this push to market, improving the accuracy of face recognition technologies remains an active area of research in academia and industry. Google’s FaceNet system, which achieved near-perfect accuracy on the Labeled Faces in the Wild dataset (Schroff et al., 2015), exemplifies one such effort. Additionally, recent advances with deep learning algorithms (Taigman et al., 2014; Parkhi et al., 2015) show much promise in strengthening the robustness of the face identification and authentication techniques used today. Indeed, state-of-the-art face identification systems can now outperform their human counterparts (Lu and Tang, 2014), and this high accuracy is one of the driving factors behind the increased use of face recognition systems.

¹ See, for example, “Apple Acquires Face Recognition, Expression Analysis firm, Emotient”, TechTimes, Jan, 2016; “Google Acquires Facial Recognition Software Company PittPar,” WSJ, 2011.

However, even given the high accuracy of modern face recognition technologies, their application in face authentication systems has left much to be desired. For instance, at the Black Hat security conference in 2009, Duc and Minh (2009) demonstrated the weaknesses of popular face authentication systems from commodity vendors like Lenovo, Asus, and Toshiba. Amusingly, Duc and Minh (2009) were able to reliably bypass face-locked computers simply by presenting the software with photographs and fake pictures of faces. Essentially, the security of these systems rested solely on the problem of face detection, rather than face authentication. This widely publicized event led to subsequent integration of more robust face authentication protocols. One prominent example is Android OS, which augmented its face authentication approach in 2012 to require users to blink while authenticating (*i.e.* as a countermeasure to still-image spoofing attacks). Unfortunately, this approach was also shown to provide little protection, and can be easily bypassed by presenting the system with two alternating images — one with the user’s eyes open, and one with her eyes closed. These attacks underscore the fact that face authentication systems require robust security features beyond mere recognition in order to foil spoofing attacks.

Loosely speaking, three types of such spoofing attacks have been used in the past, to varying degrees of success: (i) still-image-based spoofing, (ii) video-based spoofing, and (iii) 3D-mask-based spoofing. As the name suggests, still-image-based spoofing attacks present one or more still images of the user to the authentication camera; each image is either printed on paper or shown with a digitized display. Video-based spoofing, on the other hand, presents a pre-recorded video of the victim’s moving face in an attempt to trick the system into falsely recognizing motion as an indication of liveness. The 3D-mask-based approach, wherein 3D-printed facial masks are used, was recently explored by Erdogmus and Marcel (2014).

As is the typical case in the field of computer security, the cleverness of skilled, motivated adversaries drove system designers to incorporate defensive techniques in the biometric solutions they develop. This cat-and-mouse game continues to play out in the realm of face authentication systems, and the current recommendation calls for the use of well-designed face liveness detection schemes (that attempt to distinguish a real user from a spoofed one). Indeed, most modern systems

now require more active participation compared to simple blink detection, often asking the user to rotate her head or raise an eyebrow during login. Motion-based techniques that check, for example, that the input captured during login exhibits sufficient 3D behavior, are also an active area of research in face authentication.

One such example is the recent work of Li et al. (2015) that appeared in CCS'2015. In that work, the use of liveness detection was proposed as a solution to thwarting video-based attacks by checking the consistency of the recorded data with inertial sensors. Such a detection scheme relies on the fact that as a camera moves relative to a user's stationary head, the facial features it detects will also move in a predictable way. Thus, a 2D video of the victim would have to be captured under the exact same camera motion in order to fool the system.

As mentioned in (Li et al., 2015), 3D-printed facial reconstructions offer one option for defeating motion-based liveness detection schemes. In my view, a more realizable approach is to present the system with a 3D facial mesh in a virtual reality (VR) environment. Here, the motion of the authenticating camera is tracked, and the VR system internally rotates and translates the mesh to match. In this fashion, the camera observes exactly the same movement of facial features as it would for a real face, fulfilling the requirements for liveness detection. Such an attack defeats color-image- and motion-based face authentication on a fundamental level because, with sufficient effort, a VR system can display an environment that is essentially indistinguishable from real-world input.

In this paper, I show that it is possible to undermine modern face authentication systems using one such attack. Moreover, I show that an accurate facial model can be built using *only* a handful of publicly accessible photos — collected, for example, from social network websites — of the victim. From a pragmatic point of view, I am confronted with two main challenges: *i*) the number of photos of the target may be limited, and *ii*) for each available photo, the illumination setting is unknown and the user's pose and expression are not constrained. To overcome these challenges, I leverage robust, publicly available 3D face reconstruction methods from the field of computer vision, and adapt these techniques to fit my needs. Once a credible synthetic model of a user is obtained, I then employ entry-level virtual reality displays to defeat the state of the art in liveness detection.

The rest of this chapter is laid out as follows: §4.2 outlines the steps I take to perform my VR-based attack. In §4.3, I evaluate the performance of my method on five commercial face authentication systems and, additionally, on a proposed state-of-the-art system for liveness detection. I suggest steps that could be taken to mitigate my attack in §4.4.

4.2 Our Approach

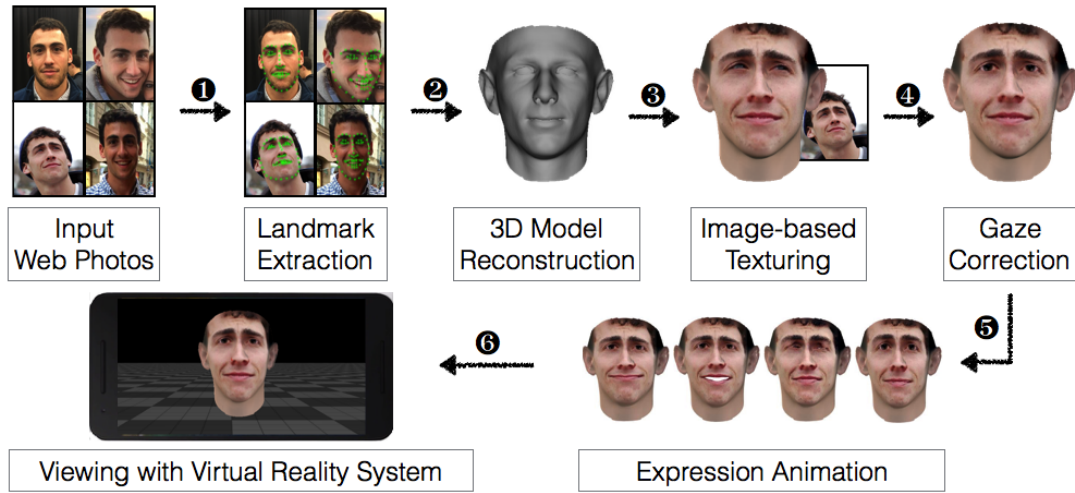


Figure 4.1: Overview of my proposed approach.

A high-level overview of my approach for creating a synthetic face model is shown in Figure 4.1. Given one or more photos of the target user, I first automatically extract the landmarks of the user’s face (stage ❶). These landmarks capture the pose, shape, and expression of the user. Next, I estimate a 3D facial model for the user, optimizing the geometry to match the observed 2D landmarks (stage ❷). Once I have recovered the shape of the user’s face, I use a single image to transfer texture information to the 3D mesh. Transferring the texture is non-trivial since parts of the face might be self-occluded (*e.g.*, when the photo is taken from the side). The texture of these occluded parts must be estimated in a manner that does not introduce too many artifacts (stage ❸). Once the texture is filled, I have a realistic 3D model of the user’s face based on a single image.

However, despite its realism, the output of stage ❸ is still not able to fool modern face authentication systems. The primary reason for this is that modern face authentication systems use

the subject's gaze direction as a strong feature, requiring the user to look at the camera in order to pass the system. Therefore, I must also automatically correct the direction of the user's gaze on the textured mesh (stage ④). The adjusted model can then be deformed to produce animation for different facial expressions, such as smiling, blinking, and raising the eyebrows (stage ⑤). These expressions are often used as liveness clues in face authentication systems, and as such, I need to be able to automatically reproduce them on my 3D model. Finally, I output the textured 3D model into a virtual reality system (stage ⑥).

Using this framework, an adversary can bypass both the face recognition and liveness detection components of modern face authentication systems. In what follows, I discuss the approach I take to solve each of the various challenges that arise in my six-staged process.

4.2.1 Facial Landmark Extraction

Starting from multiple input photos of the user, my first task is to perform facial landmark extraction. Following the approach of Zhu et al. (2015), I extract 68 2D facial landmarks in each image using the supervised descent method (SDM) (Xiong and De la Torre, 2013). SDM successfully identifies facial landmarks under relatively large pose differences (± 45 deg yaw, ± 90 deg roll, ± 30 deg pitch). I chose the technique of Zhu et al. (2015) because it achieves a median alignment error of 2.7 pixels on well-known datasets (Baker and Matthews, 2004) and outperforms other commonly used techniques (*e.g.*, (Belhumeur et al., 2013)) for landmark extraction.

For my needs, SDM works well on most online images, even those where the face is captured at a low resolution (*e.g.*, 40×50 pixels). It does, however, fail on a handful of the online photos I collected (less than 5%) where the pose is beyond the tolerance level of the algorithm. If this occurs, I simply discard the image. Example landmark extractions are shown in Figure 4.2.

4.2.2 3D Model Reconstruction

The 68 extracted 3D point landmarks from each of the N input images provide us with a set of coordinates $s_{i,j} \in \mathbb{R}^2$, with $1 \leq i \leq 68, 1 \leq j \leq N$. The projection of the 3D points $S_{i,j} \in \mathbb{R}^3$ of

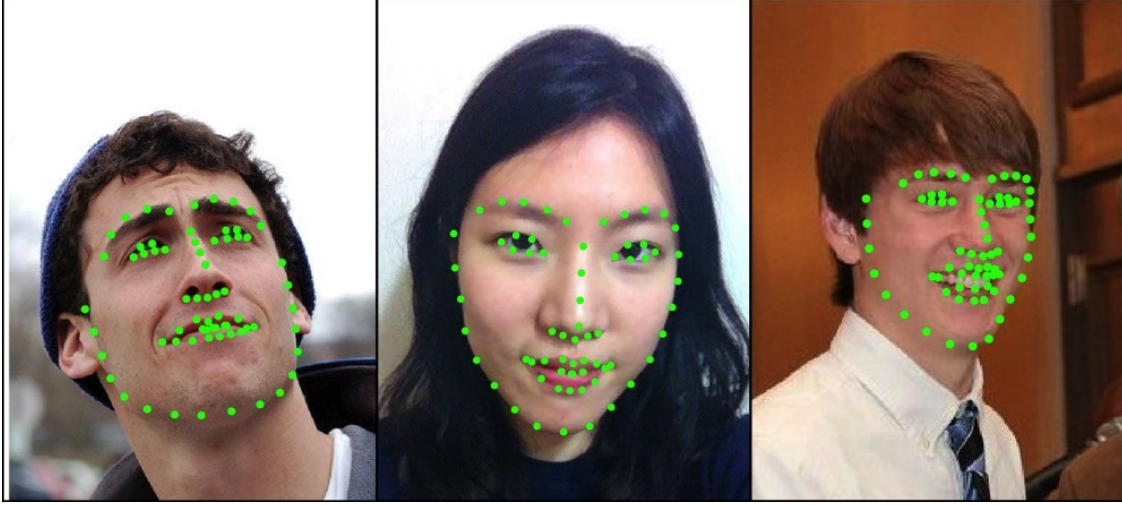


Figure 4.2: Examples of facial landmark extraction

the face onto the image coordinates $s_{i,j}$ follows what is called the “weak perspective projection” (WPP) model (Horaud et al., 1997), computed as follows:

$$s_{i,j} = f_j P R_j (S_{i,j} + t_j), \quad (4.1)$$

where f_j is a uniform scaling factor; P is the projection matrix $\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}$; R_j is a 3×3 rotation matrix defined by the pitch, yaw, and roll, respectively, of the face relative to the camera; and $t_j \in \mathbb{R}^3$ is the translation of the face w.r.t. the camera. Among these parameters, only $s_{i,j}$ and P are known, and so I must estimate the others.

Fortunately, a large body of work exists on the shape statistics of human faces. Following Zhu et al. (2015), I capture face characteristics using the 3D Morphable Model (3DMM) (Paysan et al., 2009) with an expression extension proposed by Chu et al. (2014). This method characterizes variations in face shape for a population using principal component analysis (PCA), with each individual’s 68 3D point landmarks being concatenated into a single feature vector for the analysis. These variations can be split into two categories: constant factors related to an individual’s distinct appearance (identity), and non-constant factors related to expression. The identity axes capture characteristics such as face width, brow placement, or lip size, while the expression axes capture

variations like smiling versus frowning. Example axes for variations in expression are shown in Figure 4.3.

More formally, for any given individual, the 3D coordinates $S_{i,j}$ on the face can be modeled as

$$S_{i,j} = \bar{S}_i + A_i^{id} \alpha^{id} + A_i^{exp} \alpha_j^{exp}, \quad (4.2)$$

where \bar{S}_i is the statistical average of $S_{i,j}$ among the individuals in the population, A_i^{id} is the set of principal axes of variation related to identity, and A_i^{exp} is the set of principal axes related to expression. α^{id} and α_j^{exp} are the identity and expression weight vectors, respectively, that determine *person-specific* facial characteristics and expression-specific facial appearance. I obtain \bar{S}_i and A_i^{id} using the 3D Morphable Model (Paysan et al., 2009) and A_i^{exp} from Face Warehouse (Cao et al., 2014).

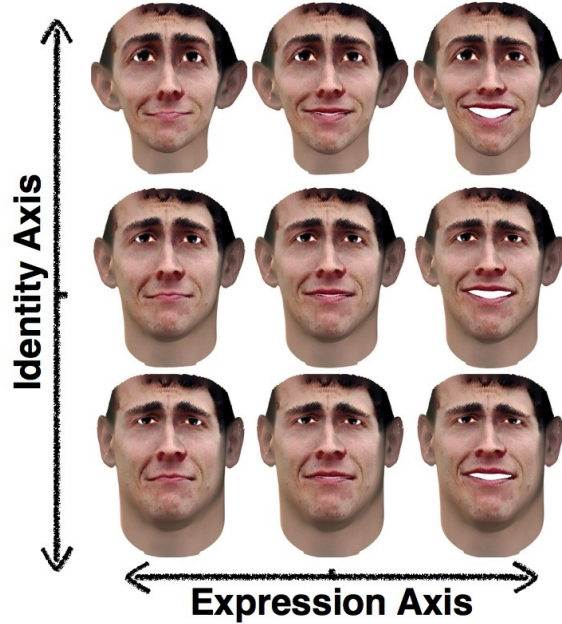


Figure 4.3: Illustration of identity axes (heavy-set to thin) and expression axes (pursed lips to open smile).

When combining Eqs. (4.1) and (4.2), I inevitably run into the so-called “correspondence problem.” That is, given each identified facial landmark $s_{i,j}$ in the input image, I need to find the corresponding 3D point $S_{i',j}$ on the underlying face model. For landmarks such as the corners of

the eyes and mouth, this correspondence is self-evident and consistent across images. However, for contour landmarks marking the edge of the face in an image, the associated 3D point on the user’s facial model is pose-dependent: when the user is directly facing the camera, their jawline and cheekbones are fully in view, and the observed 2D landmarks lie on the fiducial boundary on the user’s 3D facial model. When the user rotates their face left (or right), however, the previously observed 2D contour landmarks on the left (resp. right) side of the face shift out of view. As a result, the observed 2D landmarks on the edge of the face correspond to 3D points closer to the center of the face. This 3D point displacement must be taken into account when recovering the underlying facial model.

Qu et al. (2015a) deal with contour landmarks using constraints on surface normal direction, based on the observation that points on the edge of the face in the image will have surface normals perpendicular to the viewing direction. However, this approach is less robust because the normal direction cannot always be accurately estimated and, as such, requires careful parameter tuning. Zhu et al. (2015) proposed a “landmark marching” scheme that iteratively estimates 3D head pose and 2D contour landmark position. While their approach is efficient and robust against different face angles and surface shapes, it can only handle a single image and cannot refine the reconstruction result using additional images.

My solution to the correspondence problem is to model 3D point variance for each facial landmark using a pre-trained Gaussian distribution (see Appendix C). Unlike the approach of Zhu et al. (2015) which is based on a single image input, I solve for pose, perspective, expression, and neutral-expression parameters over *all* images jointly. From this, I obtain a neutral-expression model S_i of the user’s face. A typical reconstruction, S_i , is presented in Figure 4.4.

4.2.3 Facial Texture Patching

Given the 3D facial model, the next step is to patch the model with realistic textures that can be recognized by the face authentication systems. Due to the appearance variation across social media photos, I have to achieve this by mapping the pixels in a *single* captured photo onto the 3D facial

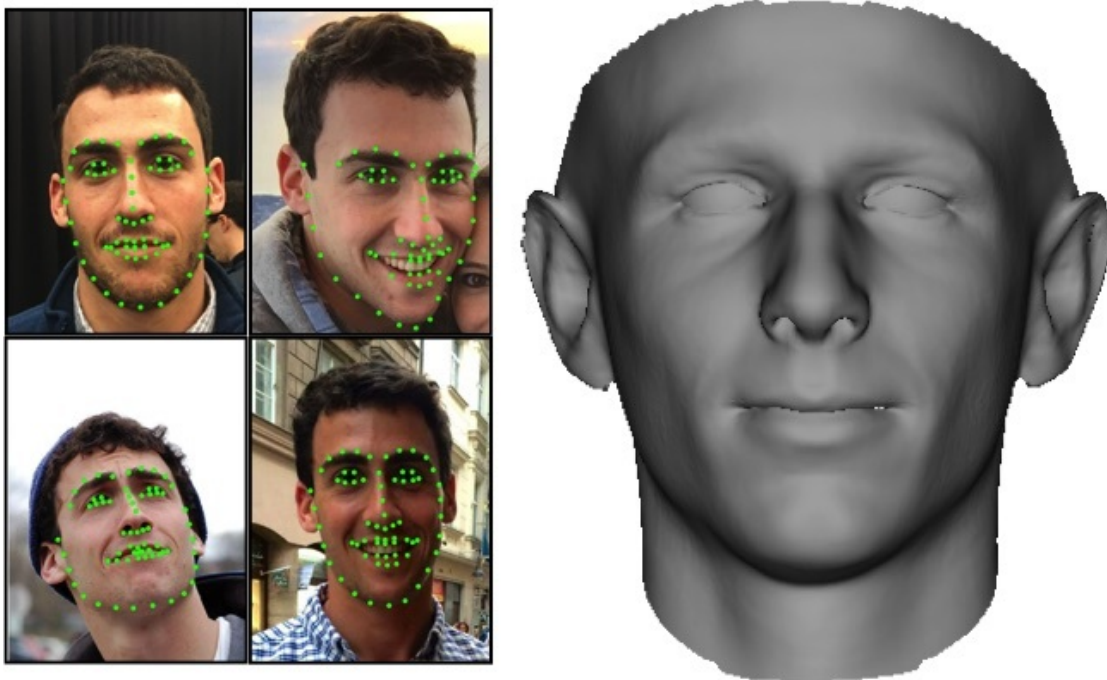


Figure 4.4: 3D facial model (right) built from facial landmarks extracted from four images (left).

model, which avoids the challenges of mixing different illuminations of the face. However, this still leaves many of the regions without texture, and those untextured spots will be noticeable to modern face authentication systems. To fill these missing regions, the naïve approach is to utilize the vertical symmetry of the face and fill the missing texture regions with their symmetrical complements. However, doing so would lead to strong artifacts at the boundary of missing regions. A realistic textured model should be free of these artifacts.

To lessen the presence of these artifacts, one approach is to iteratively average the color of neighboring vertices as a color trend and then mix this trend with texture details (Qu et al., 2015b). However, such an approach over-simplifies the problem and fails to realistically model the illumination of facial surfaces. Instead, I follow the suggestion of Zhu et al. (2015) and estimate facial illumination using spherical harmonics (Zhang and Samaras, 2006), then fill in texture details with Poisson editing (Pérez et al., 2003). In this way, the output model will appear to have a more natural illumination. Sadly, I cannot use their approach directly as it reconstructs a planar

normalized face, instead of a 3D facial model, and so I must extend their technique to the 3D surface mesh.

The idea I implemented for improving my initial textured 3D model is as follows: Starting from the single photo chosen as the main texture source, I first estimate and subsequently remove the illumination conditions present in the photo. Next, I map the textured facial model onto a plane via a conformal mapping, then impute the unknown texture using 2D Poisson editing. I further extend their approach to three dimensions and perform Poisson editing directly on the surface of the facial model. Intuitively, the idea behind Poisson editing is to keep the detailed texture in the editing region while enforcing the texture’s smoothness across the boundary. This process is defined mathematically as

$$\Delta f = \Delta g, s.t. f|_{\partial\Omega} = f^0|_{\partial\Omega}, \quad (4.3)$$

where Ω is the editing region, f is the editing result, f^0 is the known original texture value, and g is the texture value in the editing region that is unknown and needs to be patched with its reflection complement. On a 3D surface mesh, every vertex is connected with 2 to 8 neighbors. Transforming Eq. 4.3 into discrete form, I have

$$|N_p|f_p - \sum_{q \in N_p \cap \Omega} f_q = \sum_{q \in N_p \cap \bar{\Omega}} f_q^0 + (\Delta g)_p, \quad (4.4)$$

where N_p is the neighborhood of point p on the mesh. My enhancement is a natural extension of the Poisson editing method suggested in the seminal work of Pérez et al. (2003), although no formulation was given for 3D. By solving Eq. 4.4 instead of projecting the texture onto a plane and solving Eq. 4.3, I obtain more realistic texture on the facial model, as shown in Figure 4.5.

4.2.4 Gaze Correction

I now have a realistic 3D facial model of the user. Yet, I found that models at stage ③ were unable to bypass most well-known face recognition systems. Digging deeper into the reasons why, I observed that most recognition systems rely heavily on gaze direction during authentication, *i.e.* they



Figure 4.5: Naïve symmetrical patching (left); Planar Poisson editing (middle); 3D Poisson editing (right).

fail if the user is not looking at the device. To address this, I introduce a simple, but effective, approach to correct the gaze direction of my synthetic model (Figure 4.1, Stage ④).

The idea is as follows. Since I have already reconstructed the texture of the facial model, I can synthesize the texture data in the eye region. These data contain the color information from the sclera, cornea, and pupil and form a three-dimensional distribution in the RGB color space. I estimate this color distribution with a 3D Gaussian function whose three principal components can be computed as (b_1, b_2, b_3) with weight $(\sigma_1, \sigma_2, \sigma_3)$, $\sigma_1 \geq \sigma_2 \geq \sigma_3 > 0$. I perform the same analysis for the eye region of the average face model obtained from 3DMM (Paysan et al., 2009), whose eye is looking straight towards the camera, and I similarly obtain principal color components $(b_1^{std}, b_2^{std}, b_3^{std})$ with weight $(\sigma_1^{std}, \sigma_2^{std}, \sigma_3^{std})$, $\sigma_1^{std} \geq \sigma_2^{std} \geq \sigma_3^{std} > 0$. Then, I convert the eye texture from the average model into the eye texture of the user. For a texture pixel c in the eye region of average texture, I convert it to

$$c_{convert} = \sum_{i=1}^3 \frac{\sigma_i}{\sigma_i^{std}} (c' b_i^{std}) b_i. \quad (4.5)$$

In effect, I align the color distribution of the average eye texture with the color distribution of the user's eye texture. By patching the eye region of the facial model with this converted average texture, I realistically capture the user's eye appearance with forward gaze.

4.2.5 Adding Facial Animations

Some of the liveness detection methods that I test require that the user performs specific actions in order to unlock the system. To mimic these actions, I can simply animate my facial model using a pre-defined set of facial expressions (*e.g.*, from FaceWarehouse (Cao et al., 2014)). Recall that in deriving in Eq. 4.2, I have already computed the weight for the identity axis α^{id} , which captures the user-specific face structure in a neutral expression. I can adjust the expression of the model by substituting a specific, known expression weight vector α_{std}^{exp} into Eq. 4.2. By interpolating the model’s expression weight from 0 to α_{std}^{exp} , I am able to animate the 3D facial model to smile, laugh, blink, and raise the eyebrows (see Figure 4.6).



Figure 4.6: Animated expressions. From left to right: smiling, laughing, closing the eyes, and raising the eyebrows.

4.2.6 Leveraging Virtual Reality

While the previous steps were necessary to recover a realistic, animated model of a targeted user’s face, my driving insight is that virtual reality systems can be leveraged to display this model as if it were a real, three-dimensional face. This VR-based spoofing constitutes a fundamentally new class of attacks that exploit weaknesses in camera-based authentication systems.

In a VR system, the synthetic 3D face of the user is displayed on the screen of the device, and as the device rotates and translates in the real world, the 3D face moves accordingly. To an observing face authentication system, the depth and motion cues of the display exactly match what would be expected for a human face. My experimental VR setup consists of custom 3D-rendering software

displayed on a Nexus 5X smart phone. Given the ubiquity of smart phones in modern society, my implementation is practical and comes at no additional hardware cost to an attacker. In practice, any device with similar rendering capabilities and inertial sensors could be used.

On smart phones, accelerometers and gyroscopes work in tandem to provide the device with a sense of self-motion. An example use case is detecting when the device is rotated from a portrait view to a landscape view, and rotating the display, in response. However, these sensors are not able to recover absolute *translation* — that is, the device is unable to determine how its position has changed in 3D space. This presents a challenge because without knowledge of how the device has moved in 3D space, I cannot move my 3D facial model in a realistic fashion. As a result, the observed 3D facial motion will not agree with the device’s inertial sensors, causing my method to fail on methods like that of Li et al. (2015) that use such data for liveness detection.

Fortunately, it is possible to track the 3D position of a moving smart phone using its outward-facing camera with structure from motion (see §2.3.3.2). Using the camera’s video stream as input, the method works by tracking points in the surrounding environment (*e.g.*, the corners of tables) and then estimating their position in 3D space. At the same time, the 3D position of the camera is recovered relative to the tracked points, thus inferring the camera’s change in 3D position. Several computer vision approaches have been recently introduced to solve this problem accurately and in real time on mobile devices (Schops et al., 2015; Kolev et al., 2014; Tanskanen et al., 2013; Ventura et al., 2014). In my experiments, I make use of a printed marker² placed on a wall in front of the camera, rather than tracking arbitrary objects in the surrounding scene; however, the end result is the same. By incorporating this module into my proof of concept, the perspective of the viewed model due to camera translation can be simulated with high consistency and low latency.³

An example setup for my attack is shown in Figure 4.7. The VR system consists of a Nexus 5X unit using its outward-facing camera to track a printed marker in the environment. On the Nexus

² See Goggle Paper at <http://gogglepaper.com/>

³ Specialized VR systems such as the Oculus Rift could be used to further improve the precision and latency of camera tracking. Such advanced, yet easily obtainable, hardware has the potential to deliver even more sophisticated VR attacks compared to what is presented here.

5X screen, the system displays a 3D facial model whose perspective is always consistent with the spatial position and orientation of the authentication device. The authenticating camera views the facial model on the VR display, and it is successfully duped into believing it is viewing the real face of the user.

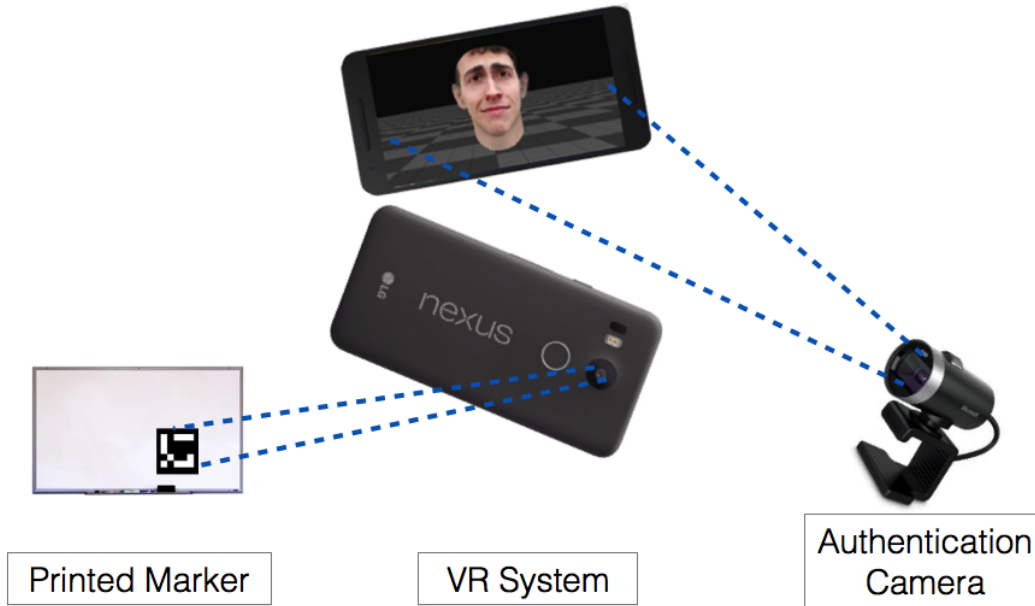


Figure 4.7: Example setup using virtual reality to mimic 3D structure from motion. The authentication system observes a virtual display of a user’s 3D facial model that rotates and translates and the device moves. To recover the 3D translation of the VR device, an outward-facing camera is used to track a marker in the surrounding environment.

4.3 Evaluation

We now demonstrate that my proposed spoofing method constitutes a significant security threat to modern face authentication systems. Using real social media photos from consenting users, we successfully broke five commercial authentication systems with a practical, end-to-end implementation of my approach. To better understand the threat, we further systematically run lab experiments to test the capabilities and limitations of my proposed method. Moreover, we successfully test my proposed approach with the latest motion-based liveness detection approach by Li et al. (2015), which is not yet available in commercial systems.

Participants

We recruited 20 volunteers for my tests of commercial face authentication systems. The volunteers were recruited by word of mouth and span graduate students and faculty in two separate research labs. Consultation with my IRB departmental liaison revealed that no application was needed. There was no compensation for participating in the lab study. The ages of the participants range between 24 and 44 years, and the sample consists of 6 females and 14 males. The participants come from a variety of ethnic backgrounds (as stated by the volunteers): 6 are of Asian descent, 4 are Indian, 1 is African-American, 1 is Hispanic, and 8 are Caucasian. With their consent, we collected public photos from the users' Facebook and Google+ social media pages; we also collected any photos we could find of the users on personal or community web pages, as well as via image search on the web. The smallest number of photos we collected for an individual was 3, and the largest number was 27. The average number of photos was 15, with a standard deviation of approximately 6 photos. No private information about the subjects was recorded beside storage of the photographs they consented to. Any display of images of subjects displayed in this chapter was done with the consent of that particular volunteer.

For my experiments, we manually extracted the region around the user's face in each image. An adversary could also perform this action automatically using tag information on social media sites, when available. One interesting aspect of social media photos is they may capture significant physical changes of users over time. For instance, one of my participants lost 20 pounds in the prior 6 months, and my reconstruction had to utilize images from before and after this change. Two other users had frequent changes in facial hair styles – beards, moustaches, and clean-shaven – all of which we used for my reconstruction. Another user had only uploaded 2 photos to social media in the past 3 years. These variations all present challenges for my framework, both for initially reconstructing the user's face and for creating a likeness that matches their current appearance.

Industry-leading Solutions

We tested my approach on five advanced commercial face authentication systems: KeyLemon⁴, Mobius⁵, True Key (Intel Security, 2015), BioID (Jesorsky et al., 2001), and 1U App⁶. Table 4.1 summarizes the training data required by each system when learning a user’s facial appearance, as well as the approximate number of users for each system, when available. All systems incorporate some degree of liveness detection into their authentication protocol. KeyLemon and the 1U App require users to perform an action such as blinking, smiling, rotating the head, and raising the eyebrows. In addition, the 1U App requests these actions in a random fashion, making it more resilient to video-based attacks. BioID and True Key are motion-based systems and detect 3D facial structure as the user turns their head. Mobius checks for 3-dimensionality by detecting motion differences between the foreground face and the background scene.

Methodology

System	Training Method	# Installs
KeyLemon ³	Single video	~100,000
Mobius ²	10 still images	18 reviews
True Key ¹	Single video	50,000-100,000
BioID ²	4 videos	unknown
1U App ¹	1 still image	50-100

Table 4.1: Summary of the face authentication systems evaluated. The second column lists how each system acquires training data for learning a user’s face, and the third column shows the number approximate number of installations or reviews each system has received according to (1) the Google Play Store, (2) the iTunes store, or (3) softpedia.com. BioID is a relatively new app and does not yet have customer reviews on iTunes.

All participants were registered with the 5 face authentication systems under indoor illumination. The average length of time spent by each of the volunteers to register across all systems was 20 minutes. As a control, we first verified that all systems were able to correctly identify the users

⁴ <http://www.keylemon.com>

⁵ <http://www.biomics.com>

⁶ <http://www.1uapps.com>

in the same environment. Next, before testing my method using textures obtained via social media, we evaluated whether my system could spoof the recognition systems using photos taken in this environment. We thus captured one front-view photo for each user under the same indoor illumination and then created their 3D facial model with my proposed approach. We found that these 3D facial models were able to spoof each of the 5 candidate systems with a 100% success rate, as shown in the second column of Table 4.2

Following this, we reconstructed each user’s 3D facial model using the images collected from public online sources. As a reminder, any source image can be used as the main image when texturing the model. Since not all textures will successfully spoof the recognition systems, we created textured reconstructions from all source images and iteratively presented them to the system (manually selected in order of what we believed to be the best reconstruction, followed by the second best, and so on) until either authentication succeeded or all reconstructions had been tested.

Findings

We summarize the spoofing success rate for each system in Table 4.2. Except for the 1U system, all facial recognition systems were successfully spoofed for the majority of participants when using social media photos, and all systems were spoofed using indoor, frontal view photos. Out of my 20 participants, there were only 2 individuals for whom none of the systems was spoofed via the social-media-based attack.

Looking into the social media photos we collected of my participants, we observe a few trends among my results. First, we note that moderate- to high-resolution photos lend substantial realism to the textured models. In particular, photos taken by professional photographers (*e.g.*, wedding photos or family portraits) lead to high-quality facial texturing. Such photos are prime targets for facial reconstruction because they are often posted by other users and made publicly available. Second, we note that group photos provide consistent frontal views of individuals, albeit with lower resolution. In cases where high-resolution photos are not available, such frontal views can be used to accurately recover a user’s 3D facial structure. These photos are easily accessible via friends of users, as well.

Third, we note that the least spoof-able users were not those who necessarily had a low number of personal photos, but rather users who had few forward-facing photos and/or no photos with sufficiently high resolution. From this observation, it seems that creating a realistic texture for user recognition is the primary factor in determining whether a face authentication method will be fooled by my approach. Only a small number of photos are necessary in order to defeat facial recognition systems.

	Indoor	Social Media	
	Spoof %	Spoof %	Avg. # Tries
KeyLemon	100%	85%	1.6
Mobius	100%	80%	1.5
True Key	100%	70%	1.3
BioID	100%	55%	1.7
1U App	100%	0%	—

Table 4.2: Success rate for 5 face authentication systems using a model built from (second column) an image of the user taken in an indoor environment and (third and fourth columns) images obtained on users’ social media accounts. The fourth column shows the average number of attempts needed before successfully spoofing the target user.

We found that my failure to spoof the 1U App, as well as my lower performance on BioID, using social media photos was directly related to the poor usability of those systems. Specifically, we found the systems have a very high false rejection rate when live users attempt to authenticate themselves in different illumination conditions. To test this, we had 5 participants register their faces indoors on the 4 mobile systems.⁷ We then had each user attempt to log in to each system 10 times indoors and 10 times outdoors on a sunny day, and we counted the number of accepted logins in each environment for each system. True Key and Mobius, which we found were easier to defeat, correctly authenticated the users 98% and 100% of the time for indoor logins, respectively, and 96% and 100% of the time for outdoor logins. Meanwhile, the indoor/outdoor login rates of BioID and the 1U App were 50%/14% and 96%/48%, respectively. These high false rejection rates show that the two systems have substantial difficulty with the face recognition part of their authentication. As

⁷ As it is a desktop application, KeyLemon was excluded.

evidenced by the second column in Table 4.2, my method still handily defeats the liveness detection modules of these systems.

My findings also suggest that my approach is able to successfully handle significant changes in facial expression, illumination, and for the most part, physical characteristics such as weight and facial hair. Moreover, the approach seems to generalize to users regardless of gender or ethnicity. Given that it has shown to work on a varied collection of real-world data, we believe that the attack presented herein represents a realistic security threat model that could be exploited in the present day.

Next, to gain a deeper understanding of the realism of this threat, we take a closer look at what conditions are necessary for my method to bypass the various face authentication systems we tested. We also consider what main factors contribute to the failure cases of my method.

4.3.1 Evaluating System Robustness

To further understand the limitations of the proposed spoofing system, we test its robustness against resolution and viewing angle, which are two important factors for the social media photos users upload. Specifically, I answer the question: what is the minimum resolution and maximum head rotation allowed in an uploaded photo before it becomes unusable for spoofing attacks like mine? I further explore how low-resolution frontal images can be used to improve my success rates when high-resolution side-view images are not available.

4.3.1.1 Blurry, Grainy Pictures Still Say A Lot

To assess my ability to spoof face authentication systems when provided only with low-resolution images of a user’s face, I texture the 3D facial models of my sample users using an indoor, frontal view photo. This photo is then downsampled at various resolutions such that the distance between the user’s chin and forehead ranges between 20 and 50 pixels. Then, I attempt to spoof the True Key, BioId, and KeyLemon systems with facial models textured using the down-sampled photos. If I am successful at a certain resolution, that implies that that resolution leaks the user’s

identity information to my spoofing system. The spoofing success rate for various image resolutions is shown in Figure 4.8.

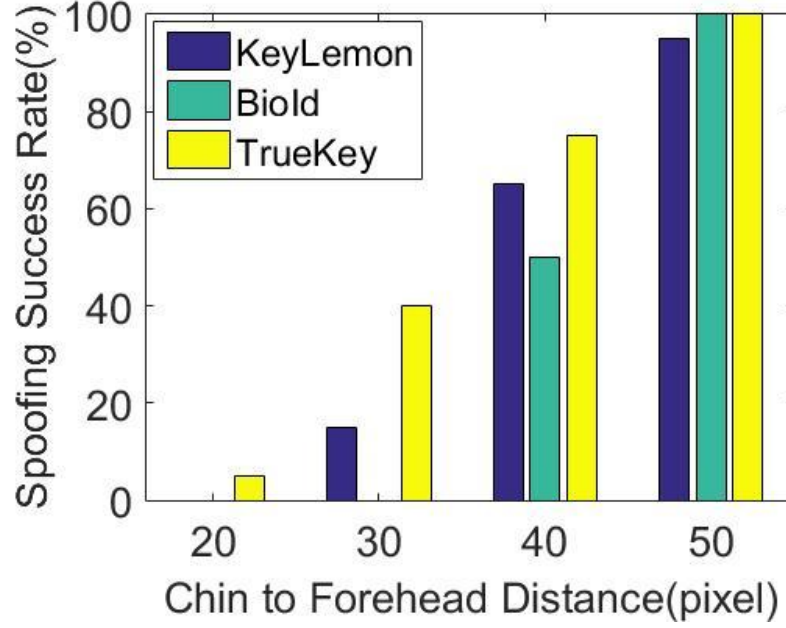


Figure 4.8: Spoofing success rate with texture taken from photos of different resolution.

The result indicates that my approach robustly spoofs face authentication systems when the height of the face in the image is at least 50 pixels. If the resolution of an uploaded photo is less than 30 pixels, the photo is likely of too low-resolution to reliably encode useful features for identifying the user. In my sample set, 88% of users had more than 6 online photos with a chin-to-forehead distance greater than 100 pixels, which easily satisfies the resolution requirement of my proposed spoofing system.

4.3.1.2 A Little to the Left, a Little to the Right

To identify the robustness of the proposed system against head rotation, I first evaluate the maximum yaw angle allowed for my system to spoof baseline systems using a single image. For all 20 sample users, I collect multiple indoor photos with yaw angle varying from 5 degrees (approximately frontal view) to 40 degrees (significantly rotated view). I then perform 3D reconstruction for each image, for each user, on the same three face authentication systems. The spoofing success rate

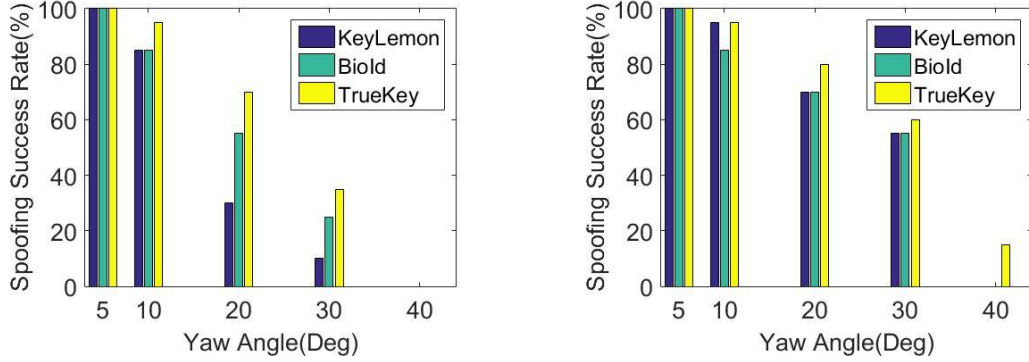


Figure 4.9: Spoofing success rate with different yaw angles. Left: Using only a single image at the specified angle. Right: Supplementing the single image with low-resolution frontal views, which aid in 3D reconstruction.

for a single input image as a function of head rotation is illustrated in Figure 4.9 (left). It can be seen that the proposed method successfully spoofs all the baseline systems when the input image has a largely frontal view. As yaw angle increases, it becomes more difficult to infer the user’s frontal view from the image, leading to a decreased spoofing success rate.

4.3.1.3 For Want of a Selfie

The results of Figure 4.9 (left) indicate that my success rate falls dramatically if given only a single image with a yaw angle larger than 20 degrees. However, I argue that these high-resolution side-angle views can serve as base images for facial texturing if additional low-resolution frontal views of the user are available. I test this hypothesis by taking, for each user, the rotated images from the previous section along with 1 or 2 low-resolution frontal view photos (chin-to-forehead distance of 30 pixels). I then reconstruct each user’s facial model and use it to spoof my baseline systems. Alone, the provided low-resolution images provide insufficient texture for spoofing, and the higher-resolution side view does not provide adequate facial structure. As shown in Figure 4.9 (right), by *using the low-resolution front views to guide 3D reconstruction and then using the side view for texturing*, the spoofing success rate for large-angle head rotation increases substantially. From a practical standpoint, low-resolution frontal views are relatively easy to obtain, since they can often be found in publicly posted group photos.

4.3.2 Seeing Your Face *Is* Enough

My approach not only defeats existing commercial systems having liveness detection — it undermines the state-of-the-art liveness detection schemes in academia. To illustrate this, I use my method to attack the recently proposed authentication approach of Li et al. (2015), which obtains a high rate of success in guarding against video-based spoofing attacks. This system adds another layer to motion-based liveness detection by requiring that the movement of the face in the captured video be consistent with the data obtained from the motion sensor of the device. Fortunately, as discussed in §4.2, the data consistency requirement is automatically satisfied with my virtual reality spoofing system because the 3D model rotates in tandem with the camera motion.

Central to Li et al. (2015)’s approach is to build a classifier that evaluates the consistency of captured video and motion sensor data. In turn, the learned classifier is used to distinguish real faces from spoofed ones. Since their code and training samples have not been made public, I implemented my own version of Li et al. (2015)’s liveness detection system and trained a classifier with my own training data. I refer the reader to (Li et al., 2015) for a full overview of the method.

Following the methodology of (Li et al., 2015), I capture video samples (and inertial sensor data) of ~ 4 seconds from the front-facing camera of a mobile phone. In each sample, the phone is held at a distance of 40cm from the subject and moved back-and-forth 20cm to the left and right. I capture 200 samples of real subjects moving the phone in front of their face, 200 samples where a pre-recorded video of a user is presented to the camera, and 200 samples where the camera is presented with a 3D reconstruction of a user in my VR environment. For training, I use a binary logistic regression classifier trained on 100 samples from each class, with the other samples used for testing. Due to the relatively small size of my training sets, I repeat my classification experiments four times, with random train/test splits in each trial, and I report the average performance over all four trials.

The results of my experiments are shown in Table 4.3. For each class (real user data, video spoof data, and VR data), I report the average number (over 4 trials) of test samples classified as real user data. I experiment with three different training configurations, which are listed in the first

Training Data	Real	Video	VR
Real+Video	98.00 / 100	1.00 / 100	99.50 / 100
Real+Video+VR	67.00 / 100	0.00 / 100	50.00 / 100
Real+VR	67.00 / 100	—	51.00 / 100

Table 4.3: Number of testing samples classified as real users. Values in the first column represent true positive rates, and the second and third columns represent false positives. Each row shows the classification results after training on the classes in the first column. The results were averaged over four trials.

column of the table. The first row shows the results when using real user data as positive samples and video spoof data as negative samples. In this case, it can easily be seen that the real-versus-video identification is almost perfect, matching the results of (Li et al., 2015). However, my VR-based attack is able to spoof this training configuration nearly 100% of the time. The second and third rows of Table 4.3 show the classification performance when VR spoof data is included in the training data. In both cases, my approach defeats the liveness detector in 50% of trials, and the real user data is correctly identified as such less than 75% of the time.

All three training configurations clearly point to the fact that my VR system presents motion features that are close to real user data. Even if the liveness detector of (Li et al., 2015) is specifically trained to look for my VR-based attack, one out of every two attacks will still succeed, with the false rejection rate also increasing. Any system using this detector will need to require multiple log-in attempts to account for the decreased recall rate; allowing multiple log-in attempts, however, allows my method more opportunities to succeed. Overall, the results indicate that the proposed VR-based attack successfully spoofs Li et al. (2015)’s approach, which is to my knowledge the state of the art in motion-based liveness detection.

4.4 Defense in Depth

While current facial authentication systems succumb to my VR-based attack, several features could be added to these systems to confound my approach. Here, I detail three such features, namely random projection of light patterns, detection of minor skin tone fluctuations related to pulse, and the use of illuminated infrared (IR) sensors. Of these, the first two could still be bypassed with

additional adversary effort, while the third presents a significantly different hardware configuration that would require non-trivial alterations to my method.

Light Projection. The principle of using light projection for liveness detection is simple: using an outward-facing light source (*e.g.*, the flashlight commonly included on camera-equipped mobile phones), flash light on the user’s face at random intervals. If the observed change in illumination does not match the random pattern, then face authentication fails. The simplicity of this approach makes it appealing and easily implementable; however, an adversary could modify my proposed approach to detect the random flashes of light and, with low latency, subsequently add rendered light to the VR scene. Random projections of structured light (Zhang et al., 2002), *i.e.* checkerboard patterns and lines, would increase the difficulty of such an attack, as the 3D-rendering system must be able to quickly and accurately render the projected illumination patterns on a model. However, structured light projection requires specialized hardware that typically is not found on smart phones and similar devices, which decreases the feasibility of this mitigation.

Pulse Detection. Recent computer vision research (Wu et al., 2012; Balakrishnan et al., 2013) has explored the prospect of video magnification, which transforms micro-scale fluctuations over time into strong visual changes. One such application is the detection of human pulse from a standard video of a human face. The method detects small, periodic color changes related to pulse in the region of the face and then amplifies this effect such that the face appears to undergo strong changes in brightness and hue. This amplification could be used as an additional method for liveness detection by requiring that the observed face have a detectable pulse. Similar ideas have been applied to fingerprint systems that check for blood flow using light emitted from beneath a prism. Of course, an attacker using my proposed approach could simply add subtle color variation to the 3D model to approximate this effect. Nevertheless, such a method would provide another layer of defense against spoofed facial models.

Infrared Illumination. Microsoft released Windows Hello as a more personal way to sign into Windows 10 devices with just a look or a touch. The new interface supports biometric authentication that includes face, iris, or fingerprint authentication. The platform includes Intel’s RealSense IR-based, rather than a color-based, facial authentication method. In principle, their approach works

in the same way as contemporary face authentication methods, but instead uses an IR camera to capture a video of the user's face. The attack presented in this chapter would fail to bypass this approach because typical VR displays are not built to project IR light; however, specialized IR display hardware could potentially be used to overcome this limitation.

One limiting factor that may make IR-based techniques less common (especially on mobile devices) is the requirement for additional hardware to support this enhanced form of face authentication. Indeed, as of this writing, only a handful of personal computers support Windows Hello.⁸ Nevertheless, the use of infrared illumination offers intriguing possibilities for the future.

⁸ See "PC platforms that support Windows Hello" for more info.

CHAPTER 5: RECONSTRUCTING TYPED INPUT FROM REPEATED REFLECTIONS

5.1 Introduction

Gone are the days when mobile phones were used exclusively for voice communication. Today, as these handheld devices have become more sophisticated, they are routinely used for a myriad of everyday activities that include checking email, text messaging, performing financial transactions, and finding directions to a location of interest. Inevitably, as my day-to-day reliance on these devices increases, the sensitive information (e.g., passwords) input on these devices becomes increasingly valuable to prying eyes.

While the academic community has long acknowledged that the ubiquity of these devices provides new opportunities for privacy abuse (as users communicate private data in ways more vulnerable to eavesdropping than ever before), the severity of the threat posed by advancements in computer vision techniques is only now being well understood (Backes et al., 2009; Raguram et al., 2011). As a case in point, both Raguram et al. (2011) and Maggi et al. (2011) recently showed that modern touch-screen smartphones may offer a greater privacy threat than their traditional counterparts. The increased risk comes from the fact that many touch-screen smartphones utilize virtual keyboards that overcome the perceived lack of tactile feedback by providing users with visual confirmation (a key “pop-out” effect) as a key is pressed. These effects, however, provide strong visual cues that can be exploited by an attacker to help identify the keys tapped on the victim’s device.

The techniques used to leverage the so-called compromising reflections in these prior works have raised my collective awareness of the realism of these threats. However, they all suffer from a similar, and profound, weakness — namely the requirement that the adversary is either within

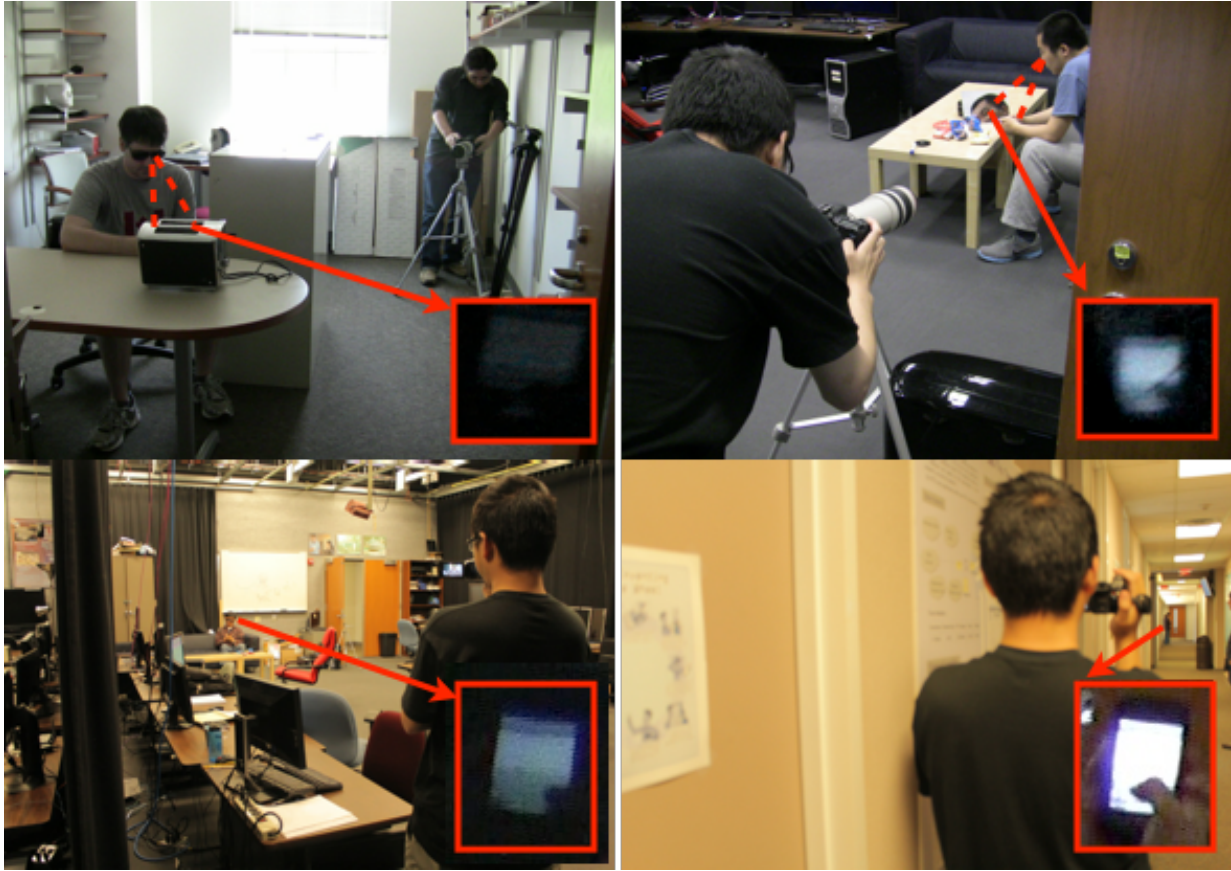


Figure 5.1: Some example threat scenarios that we investigated. Top left: Through reflection from sunglasses and toaster; Top right: Through reflection from eyeball and mirror; Bottom left: Through reflection from sunglasses; Bottom Right: Direct long-distance view. Note: As with the remaining figures in this work, this image is best viewed in color.

visual range of the victim (e.g., to ensure that the pop-out events in reflections in the victim’s sunglasses can be discerned (Raguram et al., 2011)) or is close enough to the target to avoid the use of expensive telescopes (Backes et al., 2009).

In this chapter, I push the limits of these attacks by exploiting even more fundamental, and harder to conceal, observable events. That is, unlike prior work, I do not rely on the attacker’s ability to capture detail (e.g., a key pop-out event) on the screen, but instead target a common factor in user interaction with mobile devices: the relationship between a user’s fingers and the keyboard. By tracking the positions of a user’s fingers as they move across the virtual keyboard, I can successfully reconstruct the typed input. In particular, I show that even using inexpensive consumer devices (e.g.,

small hand-held camcorders), I can successfully perform a reconstruction attack as long as both the device's orientation and the user's fingers are detectable.

Tracking fingers, rather than recognizing displayed images, broadens the class of vulnerable devices to include those without any pop-out effect, such as Apple's iPad. Moreover, my approach is capable of operating at significant distances from the victim (e.g., up to 50 meters away with a camcorder). Perhaps most importantly, my approach operates effectively even for *repeated reflections*, i.e., *reflections of reflections* in nearby objects. This feat allows us to reconstruct typed input from the image of a mobile phone's screen on a user's eyeball as reflected through a nearby mirror (see Figure 5.1), extending the privacy threat to include situations where the adversary is located *around a corner* from the user.

My approach operates despite low resolution images (a natural consequence of increasing the distance between target and camera) and high noise levels (due to reduced light levels in reflections of objects as compared to the originals). To overcome these challenges and achieve my goals, I extend a large body of work on object detection and tracking in the area of computer vision. Specifically, I extend existing finger tracking mechanisms to consider spatial context in order to reliably locate fingertips in the images. In addition, I propose a novel method for identifying fingertip trajectories based on robust estimation.

5.2 Automated Transcription

My proposed method successfully transcribes the text typed on a keyboard by exploiting video of the user typing (either directly or through up to two reflections from nearby objects). In practice, I must overcome all of the aforementioned challenges of image resolution, diffraction, and noise. To do so, I devise techniques that are resistant to low resolution, blurring, and noise. Figure 5.2 shows a high-level depiction of my approach.

I take as input a recording of the target device while the user types on its keyboard. First, I roughly estimate the location of the device in the image (Stage ❶). Next, the image of the device is aligned to a known reference template of the device's keyboard layout (Stage ❷). Then, the

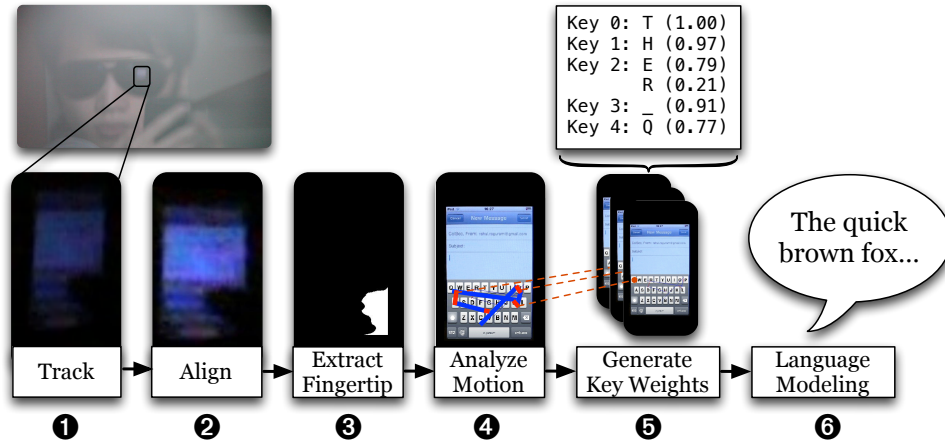


Figure 5.2: Overall design depicting the stages of my approach. First I track the motion of the mobile device (Stage ❶). Then the mobile device is aligned to a template (Stage ❷). In the stabilized image the finger is extracted (Stage ❸) and its fingertip trajectory is computed (Stage ❹). From the trajectories the likely pressed keys are identified (Stage ❺). As an optional step, I apply a language model to improve the quality of the reconstructed text (Stage ❻).

fingertips are identified in the video and the locations of the fingertips over the video frames are combined into trajectories (Stage ❸). These trajectories are then analyzed to identify the pressed keys (Stage ❹). From these pressed keys I then reconstruct the typed text (Stage ❺). Finally, as an optional post-processing step, I apply a language model in order to improve the readability of the final text (Stage ❻). Aside from some initial input in Stage ❶ to identify the first frame containing the target device, the entire process is fully automated. In what follows, I discuss each step in turn.

Stage ❶: Tracking

With a recording in hand, I first identify the device (phone, iPad, etc.) in the video so that I can focus my remaining analyses on only the relevant parts of the image. Depending on the number of reflections and the distance between the victim and the observer, the device is often only seen in a small region of each video frame (refer to Figure 5.3 for an example of reflection in an eyeball).

In order to identify the device’s location in every frame, I utilize a tracking framework based on AdaBoost (Grabner et al., 2006). The basic idea is as follows. First, the user selects the region of the first video frame (corresponding to the device) that should be localized. Given this initial position,

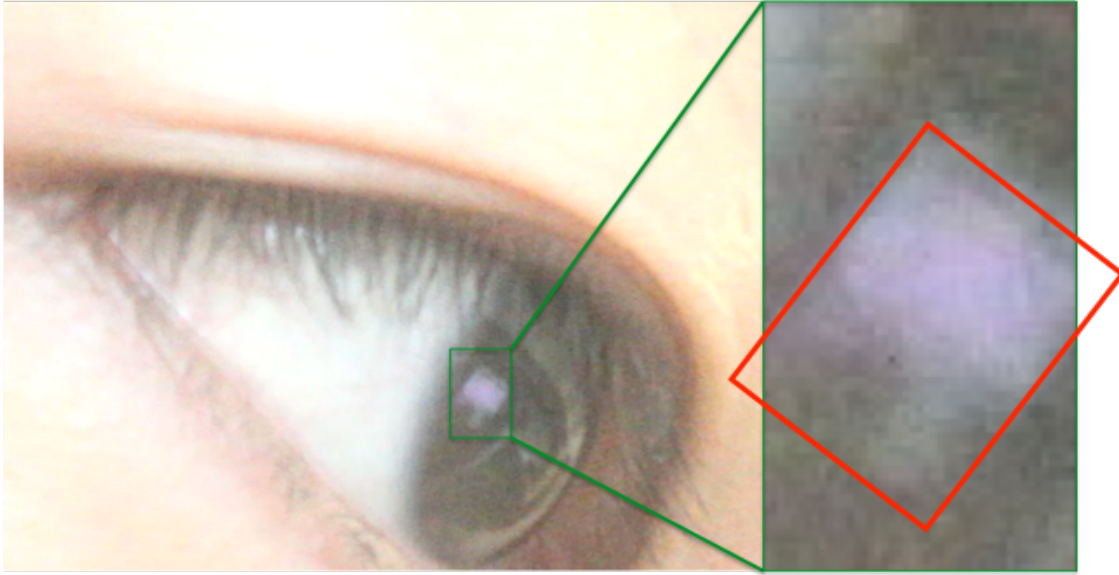


Figure 5.3: The captured image showing a reflection (of the victim’s screen) as observed in the eyeball.

AdaBoost learns the appearance of the target object, and then automatically determines its location in all succeeding video frames. Since AdaBoost tracking works relatively well even in the presence of noise, I can successfully track small or even partially occluded objects in the image. The latter is important because the device is usually occluded by the fingers as the victim types on the keyboard.

That said, despite its robustness to noise, a straightforward application of AdaBoost tracking frequently fails in my setting. This is because tracking relies on the target device maintaining a similar appearance between frames, which is not necessarily the case with the noisy images I must process. In order to mitigate this problem, I average each video frame with the temporally preceding and trailing frame in the sequence. In this manner, each image is the combination of three consecutive frames, which reduces noise as random variation will be smoothed.¹ Although averaging could cause motion blur if there is significant scene motion between the frames, I found that in practice this is not a significant issue because of the high frame rate (30 frames per second) and the small motion of both the device and the user between frames.

¹ Gaussian noise is reduced by a factor of $\sqrt{3}$.

Stage ②: Alignment

Stage ① acquired the rough position of the device in each frame. The task in Stage ② is to determine the exact location of each key in the video frame (so that later stages can identify the keys which were most likely pressed). In general, the device will have varying orientation, scale, and image position in the video. In order to determine a mapping between the image in the video and a reference keyboard layout of the device (lifted, for example, from a manual), I must estimate the transformation that will align the device in the video to a reference image of the device’s layout.

Prior work (Raguram et al., 2011; Maggi et al., 2011) faced a similar challenge, and used salient points in the image as visual features to determine the transformation. In my setting, however, those visually distinct salient feature points are no longer visible. Thus I must overcome this challenge in a different manner. The approach I take is to utilize a line detection technique that provides a more robust set of features to achieve the same goal.

Preliminary Alignment: As an initial step, I first reduce the noise present in the images by employing anisotropic filtering (Weickert, 1998). This technique (detailed in Appendix D) can be used to intelligently preserve line structure in the image (otherwise often lost in normal noise reduction techniques), while simultaneously suppressing noise in the remaining parts of the image.

To determine the correct orientation of the device, I transform the video image so that all of the device’s lines become either horizontal or vertical. This is accomplished by detecting edges within the image (Duda and Hart, 1972), and then using the lines to vote for the device’s orientation using a Hough transform. The Hough transform is a voting scheme, where each detected line in the image votes for all possible orientations of the device that conform with the line’s own orientation. For robustness, my voting scheme exploits the fact that lines of the same orientation will have a common vanishing point; hence each line is voting for its corresponding vanishing point (see Appendix E). Then, the vertical and horizontal vanishing points with the most votes represent the dominant vertical and horizontal line directions in the image. With this information at hand, I can successfully transform the device to its proper orientation.

Note, however, that even with the correct orientation, the image is still not aligned with the reference keyboard layout (i.e., the template) as the size and position may still differ. To handle this, I once again turn to a Hough transformation to vote on the translation and scale for aligning the image to the reference image of the device. The voting process relies on correspondences between the detected lines in the current frame and the lines in the template. A descriptor is built for each line in the current frame and the reference image, representing the appearance of each line's surroundings. The lines then vote for possible translation and scaling settings, and their votes are weighted by the similarity of the lines' descriptors. This voting results in a translation and scale in each of the x and y directions. Although the scale should be equal in both directions, I allow different scales (allowing affine transformation) to overcome small artifacts of slightly misaligned rotations. Appendix F details the computed transformation.

Refinement: A result of the Hough transform above is that the voting space will be quantized, leading to small residual misalignments. To provide an alignment that is stable across multiple frames and is as precise as possible I include a final refinement step (termed dense alignment (Baker and Matthews, 2004; Szeliski, 2006)). As opposed to relying on only lines, this step considers all pixels in the image to compute the final alignment. I employ a non-linear optimization in the parameters of the alignment (rotation, scale, translation) using the sum of squared differences between the overlapping pixels of the current and the first frame as an error metric for the residual misalignment. The final alignments are more stable and accurate (to sub-pixel precision) and are thus better suited for further analysis. An example alignment is shown in Figure 5.4.

Stage ③: Fingertip Extraction

At this point, the images are now aligned to the reference layout, and I am now primarily concerned with identifying the fingers as they interact with the device.

Similar to Jin et al. (2007) and Nguyen et al. (2009), I leverage Gaussian modeling of the appearance of the fingers as well as the appearance of the device. Intuitively, at this point, I

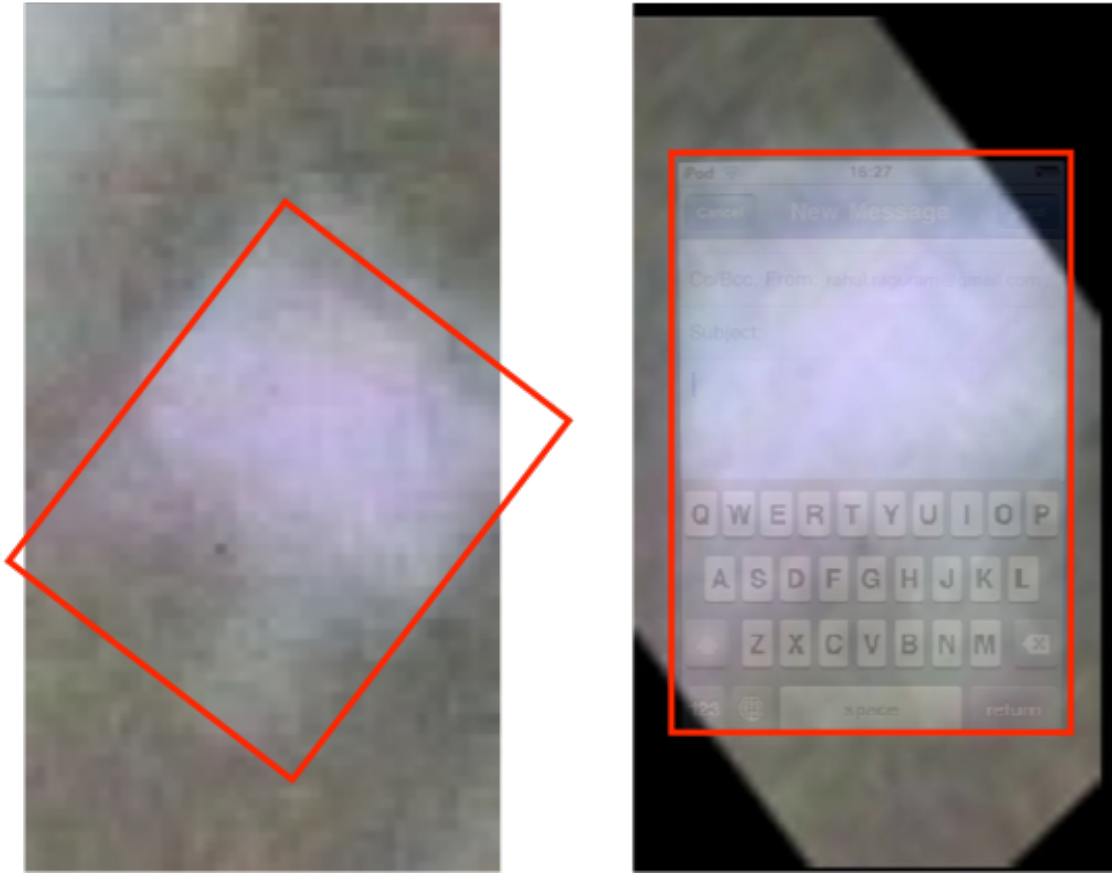


Figure 5.4: Cropped image (left) and alignment result (right) from Stage ② (overlaid with a reference device layout).

consider the keyboard area in my input image as consisting primarily of two main colors: a lighter color corresponding to the bright screen, and a darker color corresponding to the user's fingers. Gaussian modeling allows us to represent their respective appearance by two different Gaussian distributions. The idea is to assign each pixel in the keyboard area of the image to either one of the two distributions based on whichever has the highest likelihood for the pixel under consideration. To learn the different distributions, I analyze the pixel colors of the first $n = 50$ frames.

In essence, by assigning each pixel to one of the two Gaussian distributions I convert the input image into a binary image (e.g., the right image in Figure 5.5). This strategy effectively segments the fingers and the device's screen and allows us to isolate the position and orientation of the fingers. To isolate the position, I simply model the user's finger as an ellipse, and then attempt to fit an ellipse to the boundary of the segmentation. In practice, this means that I must identify points of

high curvature along the boundary of the finger's segmentation and then apply an ellipse-fitting regression function to determine the best fit (see Figure 5.5).

To identify the location of the tip of the finger, I assume that the fingertip corresponds to one of the four apexes of the ellipse. In practice, only three of the four apexes are reasonable fingertip locations because the finger only covers the keyboard from the left, right, or bottom. To determine which of the three cases is present, I consider the intersection of the ellipse with the device's boundary and select the appropriate apex. Specifically, if the ellipse intersects with the left and bottom edges of the device, then I assume that the finger is moving into view from the left; if the ellipse intersects with the right and bottom, it moves in from the right, and if the ellipse only intersects with the bottom, then it moves into view from the bottom. Notice that by repeating the above process several times (and ignoring already identified fingers) I can detect multiple fingers on the screen and locate the fingertip for each.

Stage ④: Fingertip Motion Analysis

I now turn my attention to how I identify the relative position of the fingers when keys are pressed. I take advantage of the insight that when one presses a key, the finger typically hovers at a certain position or suddenly changes direction in order to move to the next key. That observation allows us to use the location of the fingertip to identify when a key is pressed. To infer the series of likely keys pressed as the victim types, I combine the locations of the fingertips extracted in Stage ③ into trajectories for each fingertip. The trajectory of a fingertip represents its path through time, and by analyzing this path, I can deduce when a key was pressed.

To accomplish this, I propose a robust method that represents the fingertip trajectories as a curve in 3D space (where the dimensions are the u, v location in the image, corresponding to the x - y plane, and the video frame number). When a fingertip stops moving, the location in the x - y plane will cease to change, but the frame number will still increase. Hence a stopped fingertip describes a line segment in the frame number direction that is perpendicular to the x - y plane and originates at the stopping position u, v .

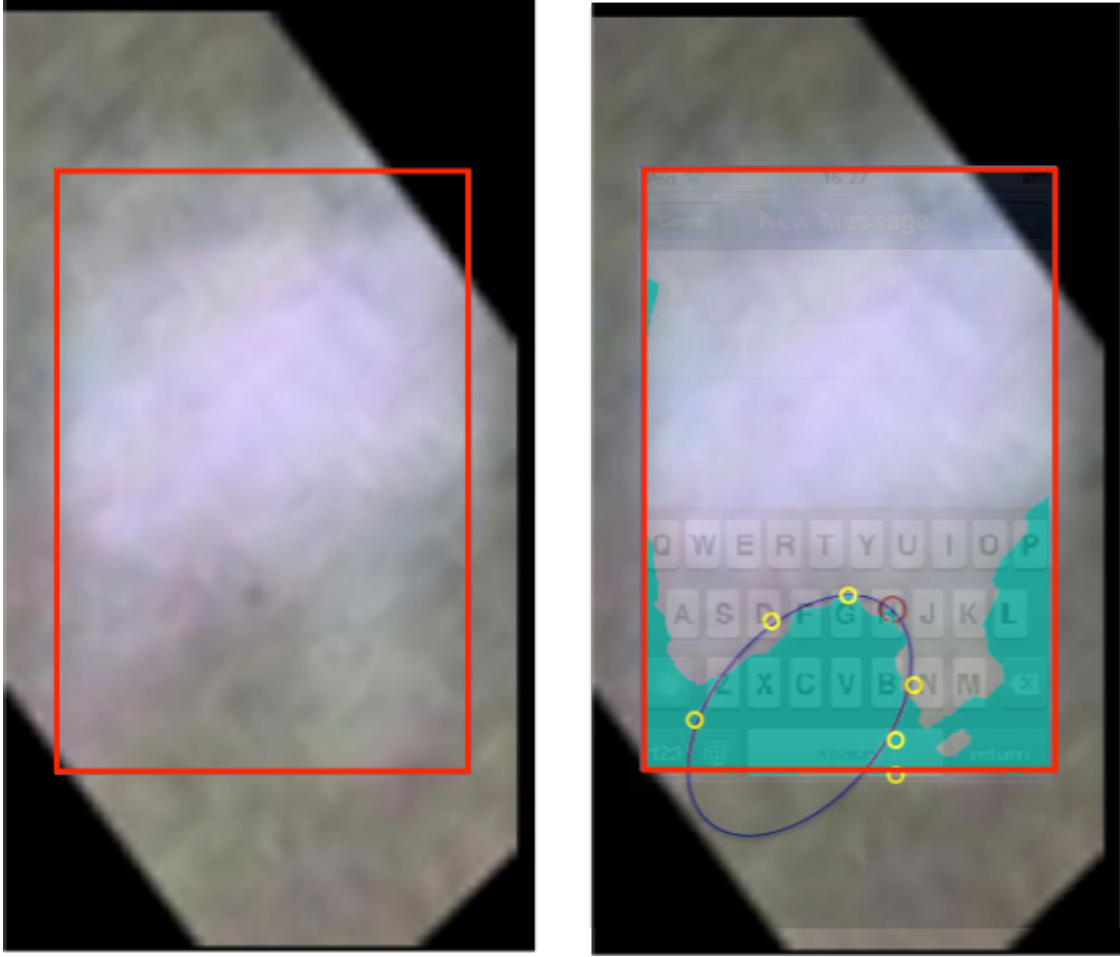


Figure 5.5: Input aligned image (left) and finger region extraction (right). Shown in the right image is the ellipse fitting (blue ellipse), segmentation (cyan), points with high curvature (yellow), detected fingertip (red circle), and an overlaid reference device layout.

To find these stopping positions or the direction changes, I approximate the entire 3D curve as a set of line segments. In order to identify the individual line segments, I use a robust model fitting technique called RANSAC (Fischler and Bolles, 1981b). Loosely speaking, the RANSAC algorithm will find a consistent model (a 3D line in this case) in the presence of noisy data even with a small number of correct data points complying with the model. Every time I run RANSAC, I model a part of the 3D curve as a line segment. Next, I remove the previously modeled part of the curve, and focus on the remaining portions. By running RANSAC repeatedly, and removing the modeled parts, I obtain a set of 3D line segments that approximate the original fingertip trajectory.

Given the set of line segments, stopping positions (u, v) are determined by line segments that are nearly perpendicular to the x - y plane. Likewise, sudden changes in direction are detected by finding two adjacent lines with a large change of direction between them. Figure 5.6 shows an example trajectory and the inferred line segment approximation.

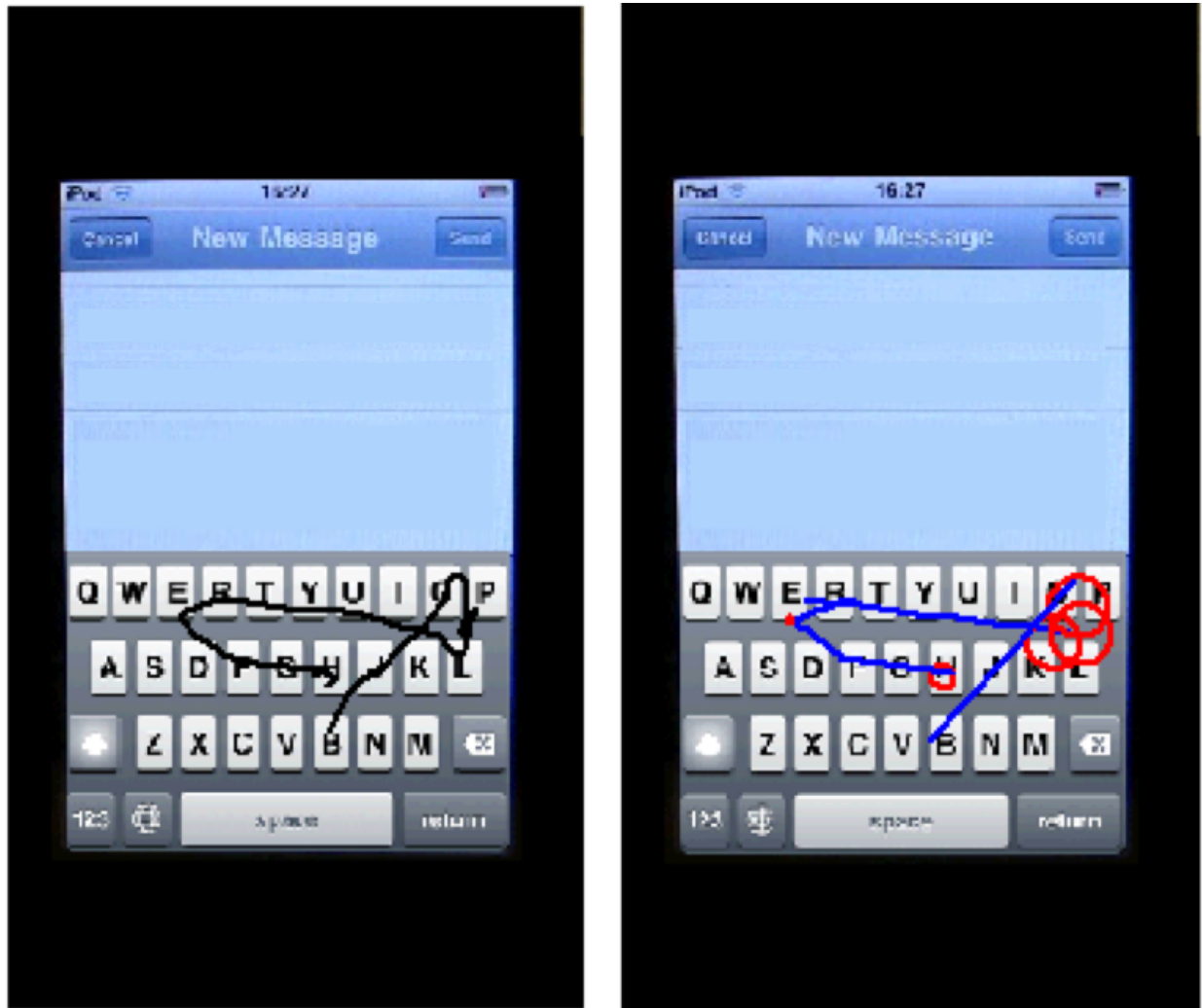


Figure 5.6: Fingertip trajectory (left) and line modeling (right). The red circles indicate stopping positions for the word “hello”, where the size of the circle corresponds to uncertainty in the stopping position.

Stage ⑤: Inferring the Keys Pressed

Given the stopping positions inferred in Stage ④, I now move on to determining what keys were likely pressed. This task is somewhat complicated by the fact that users rarely make contact

with the surface of the device with the precise tip of their finger(s). Moreover, when a user types, her fingers will not necessarily be at the same place each time they hit a particular key. Therefore, to determine which key was most likely pressed, I apply image recognition techniques to identify different positions of the user's finger with respect to the keyboard when pressing a key.

To learn the appearance of different key presses, I manually labeled 87 videos, recording the time and position of each key that was pressed. For each key on the keyboard, I collect a sample of its surrounding area as a pixel patch that is 60×100 pixels (matching the aspect ratio of a typical keyboard key). When the key under consideration is pressed, I identify the patch as a positive training sample for that key, and also use it as a negative training sample for all of the other keys on the keyboard. For each of these patches, I extract dense SIFT descriptors (Vedaldi and Fulkerson, 2010) and use these to train an AdaBoost classifier. In the end, I have a classifier for each key, where the sole purpose of the classifier is to determine whether or not its associated key has been pressed given an example pixel patch as input.

With this set of classifiers, I then analyze the key-press events detected in Stage ④. My analyses show that each event typically lasts anywhere from 10 to 20 frames, and during this time the finger stays at a relatively fixed location. I collect three adjacent frames from the middle of the duration, and extract from each a 60 by 100 pixel patch (corresponding in size to the training patch above) centered at the detected fingertip location. These patches are then fed into each of the nearby classifiers, with each classifier outputting its confidence of a key-press as a value in a range of $[0, 1]$ (with 1 being high confidence). By averaging the confidence scores of each classifier (one for each key) from the three chosen frames, I create a set of possible key-presses and their associated confidence levels. If the maximum confidence in the set falls below a predefined threshold ($\delta = 0.5$), I assume the key-press was invalid and discard the event. For a valid event with key-press confidence above δ , I record all the possible keys and their scores. By repeating this process for each key-press event, the final output of this stage is a sequence of key-press sets, where each set has a list of keys and their associated confidence values.

Note also that I do not explicitly detect presses of the “space” button, but instead treat longer pauses between key-press events as the space character. Likewise, shorter pauses are identified as the gaps between different letters in a word. Similar ideas for detecting spaces were utilized by Raguram et al. (2011) and Balzarotti et al. (2008b).

Stage ⑥: Language Model

As an optional post-processing step, I can feed the candidate keys and associated confidence values from the previous stage into a language model in order to achieve more accurate and readable results.

Similar to Raguram et al. (2011), we view the language modeling task as a *noisy channel decoding* problem and adapt techniques from the speech recognition community to perform this inference (a process known as *maximum likelihood decoding*). In particular, we employ a *cascade* of models, each of which represents a stage in the process of transforming a sequence of candidate character sets with associated confidence values into words and phrases. Each model in the cascade is represented as a *weighted finite state transducer (WFST)*, i.e., a finite state machine with an output tape in addition to the input tape, which maps sequences in one representation to sequences in another. For instance, speech recognition systems often include a component (known as the *lexicon*) which maps sequences of *phones*, the distinct sounds which comprise speech, to (valid) words. A sequence of such component models can then be used to model more complex mappings, such as that from an acoustic signal to a sequence of words and phrases, by *composition* of the WFST representations of the component models. With a lexicon as the first component, taking as input a sequence of phones, a second component (often an *n*-gram language model) maps the resulting sequence of words to likely phrases. The composition of these two models, itself a WFST, maps sequences of phones to likely phrases.

One advantage of the uniform representation of these components as WFSTs is that the *decoding* or inference step (i.e., finding the most likely phrase corresponding to the input sequence of sounds) can be performed on the composition of the component models rather than proceeding

sequentially through component models. In other words, traditional speech recognition cascades were effectively Markov chains: the output of each component depended only on the output of the previous component. Often, the resource-intensive nature of the speech recognition task forced the pruning of unlikely sequences, e.g., phones after the application of a component model. Thus a sequence of phones which might be considered unlikely (but not impossible) in the absence of a language model might be pruned in the early stages of such a speech recognition system. A WFST cascade, on the other hand, can be represented as a single WFST, which allows for inference from all components simultaneously and mitigates many problems resulting from pruning.

In our case, the input for the language modeling stage is a sequence of key-press events with character labels and associated confidence values, including marked spaces. We then employ a composite WFST model to smooth over any potential errors in the earlier stages by modifying the output words and phrases to more closely match natural English language. We first apply an edit distance model \mathcal{E} for error correction. Conceptually, the edit distance model produces, for each input string, a set of similar strings each weighted by (keyboard) edit distance from the input string. We then employ a dictionary model \mathcal{D} , which prunes those strings which do not result in dictionary words, and an n -gram language model \mathcal{L} , which promotes sequences of words that appear more frequently in English and demotes those which appear rarely. Then the cascade model can be represented as $\mathcal{C} = \mathcal{E} \circ \mathcal{D} \circ \mathcal{L}$. To find the most likely sequence of words corresponding to an input sequence of character sets (as output by the previous stage), the input is first represented as a finite state acceptor \mathcal{I} . Then the shortest path through the WFST $\mathcal{I} \circ \mathcal{C}$ gives the most likely series of English words given the input sequence of predicted character sets.

Our n -gram language model \mathcal{L} is a unigram model trained on the Brown corpus (Francis and Kucera, 1979). The dictionary model \mathcal{D} is based on the ‘medium’ word list from the Spell Checker Oriented Word Lists² with roman numerals, unusual abbreviations, and proper nouns removed. For the keyboard edit distance, we define the distance between two keys as the normalized product of the difference in rows (on a ‘vertical’ axis through Q and Z) plus one and the difference in keys (on

² wordlist.sourceforge.net

Device Name	Focal Length	Aperture	Sensor Size	Resolution
Canon 60D DSLR, 400mm Lens	400mm	F/5.6	$22.3 \times 14.9\text{mm}$	5184×3456
Canon VIXIA HG21 Camcorder	57mm	F/3.0	$4.84 \times 3.42\text{mm}$	1920×1080

Table 5.1: Specifications of the cameras used in my evaluation.

a ‘horizontal’ axis through A and L) plus one. For instance, the (unnormalized) distance between A and B is $(1 + 1) * (4 + 1) = 10$, since A and B are a single row apart vertically and four keys apart horizontally. We offset by one in each case to avoid zero values if the keys are in the same row or column.

5.3 Evaluation

Recall that one of the key motivating scenarios driving my design is the ability to reconstruct the typed input from repeated reflections. From an adversarial point of view, I am also interested in understanding the realism of the threat posed by an adversary performing such an attack from as far as possible, yet relying on inexpensive equipment. In what follows, I provide an analysis of my results under these conditions.

For the case of multiple reflections, I make use of a Canon 60D digital SLR (\$700) with 400mm Lens (\$1340). For experiments with direct line-of-sight and a single reflection, I use a Canon VIXIA HG21 Camcorder (\$1000). These devices are considerably smaller than the telescopes used previously (Backes et al., 2008, 2009; Kuhn and Kuhn, 2003) and are also more affordable. The specifications are listed in Table 5.1.

For my lab-based evaluation, five different subjects took part in my experiments. To explore different typing conditions, I asked the subjects to perform 3 different styles of typing: typing a specific sentence, providing a quick response to a question (e.g., as one might do when text messaging), and typing a password of their choosing. The chosen sentences are from “The Art of the War” and the questions from SMS messages collected by NUS (Chen and Kan, 2011). In all, I collected 73 responses containing 584 total words. In addition, I collected 15 passwords (each consisting of 5 to 8 lower-case characters).

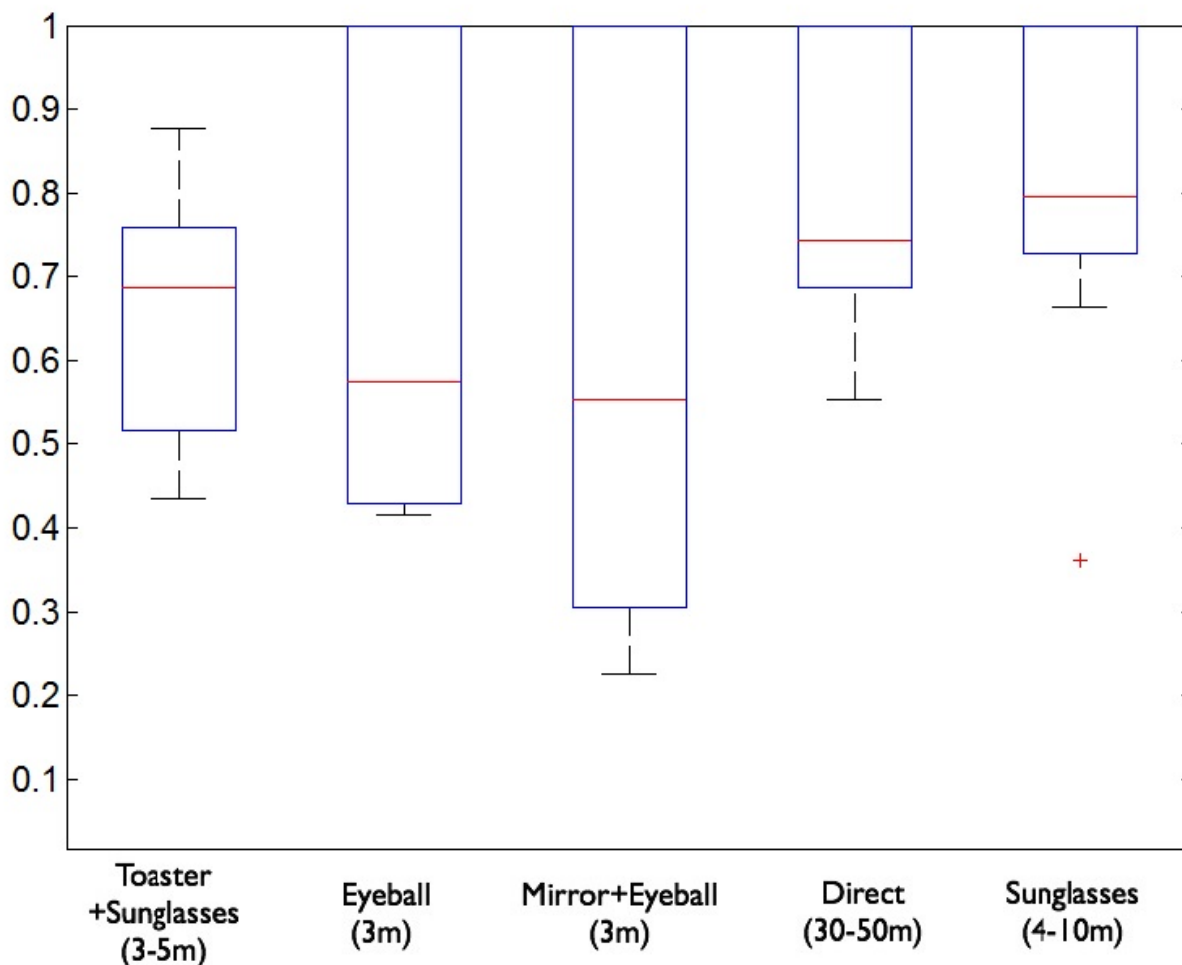


Figure 5.7: METEOR scores for my approach, broken down by scenario.

To gauge my performance, I evaluate the reconstruction results for sentences and passwords using separate metrics. The reason is obvious: the use of a natural language model can easily improve the result of the former, while for passwords a better measure is to examine how many guesses are required to recover the password given my initial hypothesis.

For evaluating the quality of my reconstructed text, I use the METEOR metric proposed by Lavie and Denkowski (2009). Essentially, METEOR scores machine translation hypotheses (i.e., my guesses) by aligning them to one or more reference translations. It estimates human perception of the readability of the sentence in terms of the adequacy and fluency of the translation. METEOR assigns scores on a scale from 0.0 to 1.0, with 1.0 indicating a perfect translation. As a guide

Scenario	Distance	Reference Sentence	Reconstructed Results	ME-TEOR Score
Toaster+Sunglasses[C1]	3m	when your round is a short one you take a walk	when your round is a short one he take a walk	0.88
	3m	when it is a long one you take a cab	when a is a long is you the a can	0.43
Mirror+Eyeball[C1]	3m	when it is a long one	when it is a long one	1.00
	3m	i am busy tonight	i at be tonight	0.33
Sunglasses[C2]	4m	if you know your enemy and you know yourself you need not fear the results of a hundred battles	if you know your enemy and you know yourself you need not fear the results of a hundred battles	1.00
	10m	if you know neither the enemy nor yourself you will succumb in every battle	if you his neither the enemy for yourself to will succumb in every battle	0.71
Direct view[C2]	30m	when your round is a short one you take a walk	when your round is a short one you take a walk	1.00
	50m	i plan to stay at home	i men to stay at home	0.74

Table 5.2: Example reconstructions. [C1]: Using Canon 60D DSLR(\$700) with 400mm Lens(\$1340). [C2]: Using Canon VIXIA HG21 Camcorder(\$1000).

for interpreting METEOR scores, Lavie (2010) suggests that scores of 0.5 or higher indicate understandable translations, while scores of 0.7 or higher indicate good or fluent translations. Examples from my experiments are shown in Table 5.2.

5.3.1 Results

The aggregate results of my approach, including confidence intervals, are illustrated in Figure 5.7. Notice that in every circumstance, my reconstruction results in an average METEOR score above 0.5, indicating an understandable level. Additionally, I reconstructed 23% (17/73) of the sentences perfectly, and 92% (67/73) of all the sentences have a METEOR score above 0.5. All of the captured videos were within the automatic focus and exposure limits of the cameras, leading

to no unusable videos which might occur in more challenging environments. I remind the reader that these results are already obtained in situations where previous efforts (Raguram et al., 2011, 2013; Maggi et al., 2011) fail, either due to an increased distance from the target or an additional reflection. These results, as well as the individual examples in Table 5.2, indicate that my attack is quite stable to changes in the distance to the target and the number and type of reflecting surfaces.

Closer look: Double Reflections

In order to evaluate the performance of double-reflection attacks I conducted the following experiments. First, I position a user with sunglasses at a desk with a nearby toaster. The user then types on an iPhone 4, and an attacker observes the typed input from a concealed location around a corner of 60 to 90 degrees by leveraging the two reflections. For the second setup, the user is once again at a desk, but instead of relying on the reflection from sunglasses, I use the reflection from the user's own eye. To enable this attack, I utilized a small mirror instead of the toaster. The use of the mirror does not influence the image quality but provides the additional ability to attack around the corner. The results without the mirror are similar (Figure 5.7). These scenarios are illustrated in Figure 5.8.

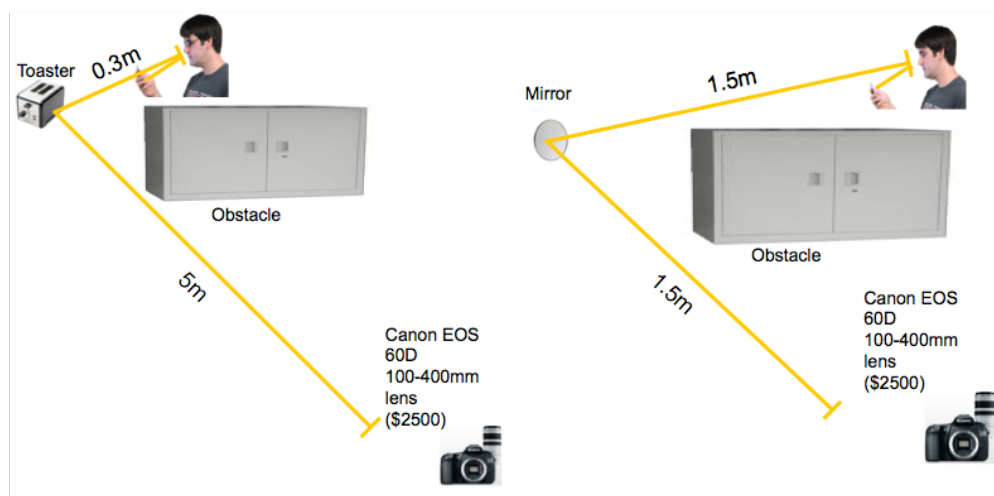


Figure 5.8: Left: the cellphone image reflects from the user's sunglasses, off the toaster's surface, and is then captured by the camera. Right: the cellphone image reflects from user's eyeball, off the mirror, and then is captured by the camera.



Figure 5.9: Example double-reflection from sunglasses and a toaster. Left: Captured image; Right: (correctly) identified keypress (T, with confidence 0.55). The brightness of the detected image (right) is adjusted for viewing convenience and has been overlaid with a reference device layout.

The use of multiple reflections naturally limits the information I can acquire from the target. With the double reflection, the main limitation of the system is the contrast between the user's fingers and the device's screen. After two reflections, the device's screen can barely be recognized by a human (see Figure 5.9). The contrast between the device screen and the background is so low that it is much harder to extract the device screen and finger regions. Yet I am still able to achieve results with an understandable level (0.65). In Figure 5.10, I illustrate the effect that contrast has on the final result. Reflections significantly reduce the contrast of the image. The RMS contrast (standard deviation of the pixel intensities) takes values in the range $[0.0, 0.5]$, where 0.0 corresponds to an image with uniform color and 0.5 corresponds to a checkerboard pattern of black and white squares. The RMS contrast in my experiments drops from around 0.35 in direct view to 0.03 with two reflections. This leads to difficulties in finger region extraction and therefore lower METEOR scores.

In the reflection off the eyeball (Figure 5.11), the finger and keyboard are even more difficult to discern. The image is both small and significantly blurred. Backes et al. (2008) discuss the theoretical boundary that a telescope with a 62 cm aperture from a distance of 2 m would be required

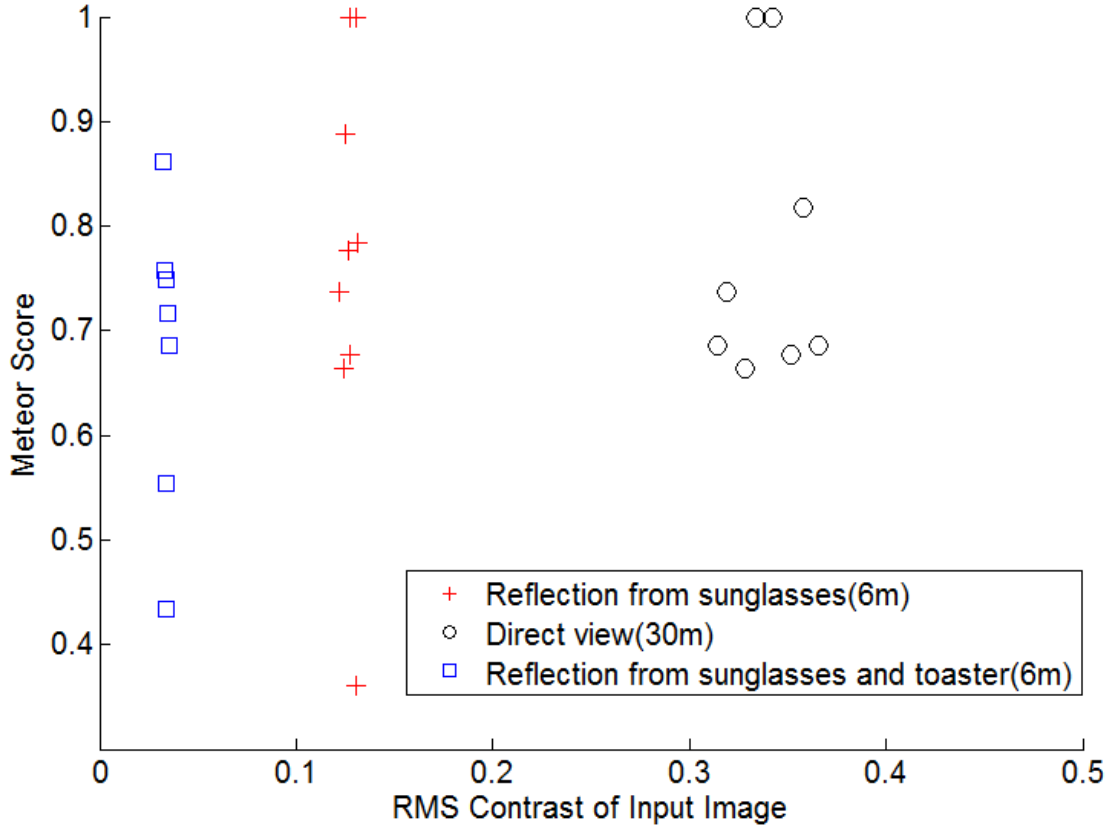


Figure 5.10: Plot of captured RMS contrast versus METEOR score for the three different scenarios (direct, single-reflection, and repeated reflections).

in order to obtain a full resolution image from the reflection of an eyeball. In my setting (i.e., a lens with a 7.1 cm aperture from a distance of 3 m), I achieve understandable reconstruction results with less than $1/10^{th}$ of the full resolution. The main problem with the reflection from the eyeball is the blurring caused by diffraction and noise, as discussed in Section 2.4.1.1, as well as partial occlusion by the eyelashes.

Closer look: Single Reflection and Direct Sight

Compared with the prior state of the art, my attack can be executed from a much farther distance, as illustrated in Figure 5.12. For example, using the same equipment (Canon VIXIA HG21 Camcorder), the approach of Raguram et al. (2013) can only run a direct attack at 24 m and a single-reflection attack at 4 m, while my work increased the distance to be 50 m and 10 m respectively (see Figure 5.7) while maintaining comparable results.

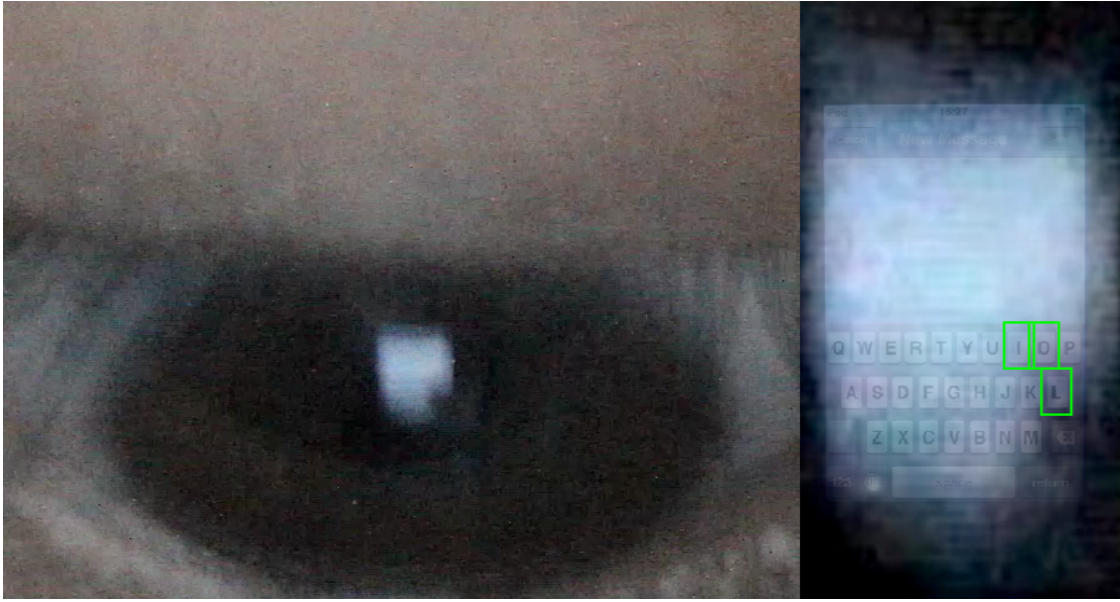


Figure 5.11: Example double-reflection from an eyeball and mirror. Left: Captured image; Right: candidate keys (I with confidence 0.10, L with confidence 0.07, and the correct key O with confidence 1.00). The right image has been overlaid with a reference device layout.

Devices without Pop-ups

As a final experiment to demonstrate that my design is applicable to a broad range of devices, I include a preliminary study on a device (a first generation iPad) with a virtual keyboard but no pop-out events. In this case, the iPad was placed at a distance of 4 meters from a pair of sunglasses which were in turn situated 4 meters from my camera. As the iPad's screen is considerably larger than the iPhone 4, it is perhaps not surprising that I am able to perfectly reconstruct all 8 sentences tested.

5.3.1.1 Password Guessing

I also apply my approach to the reconstruction of passwords. While passwords typically chosen by users may follow certain distributions that make them easier to guess, I assume that passwords are chosen with sufficient randomness to mitigate any language modeling, and therefore exclude the optional Stage ⑥ from this portion of the analysis. I instead examine the classification results from Stage ⑤ directly. One advantage of my approach is that the output of Stage ⑤ is a sequence of sets



Figure 5.12: Example single reflection from 10m. Left: Single reflection from 10m (correctly predicting I with confidence 1.0). Right: Direct view from 50m (correctly predicting T with confidence 1.0). Both images have been overlaid with a reference device layout.

of candidate keys with associated confidence values. I therefore have a natural ordering with which I can prioritize my password guessing strategy.

I implement this ordering with a best-first search, where the ‘best’ candidate password is the string which maximizes the product of the confidence values for the individual keypresses. I can calculate this product for each child of a given candidate password, where a child is identical to the candidate except that a single character has been replaced with the next most likely character for that position. Beginning with the most likely candidate, i.e., that for which the individual confidence values are maximized, I check if the candidate matches the password for which I am looking. If

Scenario	Distance	Reference (Typed)	Top Guess	Number of Guesses
Direct Sight	30m	bjтам	bjтам	1
		mstys	matya	47
		dlxmzd	elxmad	64
		zywmxq	atahxq	5967
	50m	wabjта	ewbjта	3
		tatsta	tatata	10
		wdlxmzd	wdiebae	277
Eyeball and Mirror	3m	jrairbf	jfairbf	3
		hvgcjyx	hvccnyx	620
		bgditv	cgcitv	3
Sunglasses and Toaster	3m	sjyfh	sjyfn	2
		aluhe	aaue	14
		kydhwria	kydhwria	1
		jyerpsk	jynraak	21
	5m	hdyeri	hdierx	6

Table 5.3: Expected number of guesses required to identify user passwords given a ranking, by decreasing confidence, of candidates.

not, I add each child of the candidate to a priority queue, ordered by the product of the confidence values. Once the children of one candidate are added, the candidate is discarded, the highest priority candidate is drawn from the queue, and the process begins anew.

Letters outside of the candidate set are assigned a weight of 10^{-6} , which is significantly smaller than the weights of any observed keypresses. This accounts for cases where the actual key is not contained within the predicted set. I break ties randomly, i.e., I report the average-case estimates when multiple candidates have the same value for the product of the confidence values.

Table 5.3 shows my password guessing results. Of the 15 passwords used, 6 were reconstructed in 3 or fewer guesses, 12 in less than 100 guesses, and all 15 in less than 6000 guesses. Random guessing of passwords would result in almost 6 million guesses for 5-letter passwords and almost 155 million guesses for 6-letter passwords, in expectation. Accordingly, my approach results in a speedup of four orders of magnitude. That said, I remind the reader that the passwords used were those chosen by my test subjects. Unfortunately, my test subjects chose passwords which were between 5 and 8 characters long and consisted of only lower-case letters. However, since the mobile devices in which I am interested simply overlay the lower-case keyboard with an upper-case,

Stage Name	Time Spent (sec/frame)	Percentage
Stage 1: Tracking	0.228	8.4%
Stage 2: Alignment	1.054	38.9%
Stage 3: Finger Extraction	1.042	38.4%
Stage 4: Fingertip Analysis	0.123	4.5%
Stage 5: Key Weight Generation	0.256	9.4%
Stage 6: Language Model	0.007	0.2%
Total	2.708	100.0%

Table 5.4: Runtime performance of my approach.

numeric, or symbolic keyboard as necessary, extending my analysis to cover these cases would not be difficult.

5.3.1.2 Runtime Performance

The overall performance is 0.37 frames per second, the composition of which is shown in Table 5.4. The current version of my approach is mainly implemented in MATLAB; only the tracking scheme in Stage ❶ is optimized with C++ and the use of a GPU. As such, re-implementation of the remainder of my approach in C++ and/or on the GPU would likely result in a significant speedup.

5.4 Mitigations

My attack invalidates a significant portion of prior defences (e.g., like eliminating key pop-up events (Raguram et al.)). Perhaps the most natural mitigation is to apply a privacy screen/film to the device. This will limit the amount of light emitted, making reconstruction of reflected images (with inherently reduced brightness) extremely difficult. Moreover, it will significantly limit the possibility of direct sight attacks by narrowing the angle of screen observation. However, many materials are diffuse reflectors (reflection in all directions), which may lift the restriction on viewing angle imposed by privacy films by allowing an adversary to exploit the reflection rather than a direct view of the screen.

More esoteric mitigations that are focused on hindering the recovery of sensitive input (e.g., passwords) call for the application of gaze-based passwords (Kumar et al., 2007; Weaver et al.,

2011), shoulder-surfing resistant graphical password schemes (e.g., (Sobrado and Birget, 2002; Hoanca and Mock, 2008)), and randomized keyboards (Tan et al., 2005; Zhang et al., 2012). Gaze-based password systems, for example, take into consideration the focus of the user's eyes on the screen as the source of input, greatly reducing the ability of an adversary without an unobstructed view of the user's eyes and orientation with respect to the screen (Kumar et al., 2007; Weaver et al., 2011). Likewise, randomized keyboards permute on-screen keyboard layouts as the user enters her password (Tan et al., 2005; Zhang et al., 2012), effectively thwarting shoulder-surfing attacks during such times.

Clearly, these proposals would hamper the ability to reconstruct entered passwords using approaches similar to mine; in particular, attacking either the graphical password schemes or the randomized keyboard schemes would require the ability to discern detail on the screen of the device. That said, all of these proposals are limited to password entry, and as such, only offer a partial solution to this challenge of limiting recovery of typed input exposed via compromising reflections. Moreover, the real-world usability of these proposals remains unclear.

5.5 Conclusions

In this chapter, I provide a technique for accurately reconstructing the text typed on a mobile device, with a broader range of credible threats that underscore the realism of the attack. Specifically, I show that it is possible to perform such attacks in a number of challenging scenarios: on devices with any type of virtual or physical keyboard, without direct line-of-sight, and at distances farther away from the victim than previously thought possible. My attack can even directly use reflections on the eye-ball, which was not possible before due to the noise and physical boundaries of the optics in prior work. My empirical analysis, which covers a number of scenarios (including direct line-of-sight, single reflections, and repeated reflections) as well as a range of distances (3 m - 50 m), demonstrates the success of my approach and underscores the significance of this privacy threat.

CHAPTER 6: INFERRING TV CONTENT FROM LIGHT EFFUSIONS

6.1 Introduction

To most of us, it would come as no surprise that much of the population is addicted to watching television, due in part to the wide variety of entertainment (e.g., reality TV, game shows, movies, premium channels, political commentary, 24hr news, etc.) that is offered in today's competitive market place — be that online or via broadcast TV. Indeed, so-called streaming TV and Internet connectivity now liberate viewers from restrictive schedules, making watching shows part of a wider and richer experience in homes. Admittedly, although the physical TV sets of the past are not as popular as they once were, TV is here to stay and its role in delivering compelling viewing experiences will continue for decades.

The markedly richer content offered today has helped sustain living room screens as a dominant communication medium — both collectively (e.g., for watching a big game or season finale) and individually (e.g., for accessing specific content on demand). In fact, even though consumer viewing habits have undergone change in recent years (e.g., phone, tablet and computer viewing habits have steadily increased), nearly every U.S. home still owns at least one TV and 67% of Americans regularly watch television while having dinner (Gomery, 1993). The flickering lights of the scenes that play out on these TVs are easy to see when one walks through the street at nights. Yet, many of us may not have given a second thought to the amount of information these flickering patterns (caused by changes in brightness) might reveal about the programs we watch.

My findings, however, suggest that these compromising emissions of changes of brightness provide ample opportunity to confirm what specific content is being watched on a remote TV screen, even from great distances outside the home. The key intuition behind why this threat to privacy is possible lies in the fact that much of the content we watch induces flickering patterns that uniquely

identify a particular broadcast once a suitable amount of light emissions (i.e., on the order of a few minutes) has been recorded by the adversary. This surprisingly effective attack has significant privacy implications given that the video and TV programs that people watch can reveal a lot of information about them, including their religious beliefs, political view points or other private interests. For that reason, subscribers' viewing habits are considered sensitive under the U.S. Video Protection Privacy Act of 1998, which states that an individual's video viewing records must be kept confidential. Recently, a popular electronics firm came under investigation when it was revealed that its Smart TV was surreptitiously sending information on viewing habits back to the parent company in an effort to "deliver more relevant advertisements" ¹.

While the observations I leverage in this chapter are just a demonstration of potential capabilities, to the best of my knowledge, I present the first automated, end-to-end, approach for realizing the attack. Undoubtedly, the academic community has long acknowledged that video viewing records are vulnerable to different attacks (e.g., due to electromagnetic or power line behavior (Enev et al., 2011; Greveler et al., 2012; Kuhn, 2013)), but these attacks have not received widespread attention because they require access to smart power meters and other specialized equipment in order to capture the required signal. Moreover, because these attacks rely on specific TV/computer screen electronic properties they remain difficult to implement in practice.

In this chapter, I push the boundaries of these attacks by exploiting compromising emissions which are far easier to capture. In fact, I do not rely on the adversary's ability to capture an image of the screen, or its reflection on a nearby surface (e.g., (Backes et al., 2008; Xu et al., 2013)). Instead, my attack works by analyzing the changes in brightness in a room where the content is being watched, and matching the captured signal in real-time with reference signals stored in a large database. The attack can be successfully carried out with inexpensive consumer devices (e.g., web cameras, digital SLRs) and is successful when illumination changes caused by the TV screen are perceptible to the camera's sensor.

¹ See J. Brookman, *Eroding Trust: How New Smart TV Lacks Privacy by Design and Transparency* at <http://www.privacyassociation.org/>, Nov. 2013.

To ensure that the attack is resilient to noise (e.g., from a passing vehicle, the turning on/off of a light switch, or from human movement), my approach focuses squarely on significant changes in the captured sequence, instead of directly leveraging all of the captured signal. Said another way, I exploit temporal brightness information that is not adversely affected by device-specific or environmental conditions. These environmental conditions (e.g., reflections off a wall) might result in a weakened and distorted overall signal, but the temporal information of significant intensity changes will remain largely intact.

A key contribution in this chapter lies in the techniques I use to take advantage of temporal information to find matches among reference and captures signals, even in the face of significant noise and signal distortions. To do so, I extend traditional correlation measures to utilize temporal information when computing similarity scores between sequences. The resulting strategy significantly outperforms traditional correlation measures (e.g., (Greveler et al., 2012)), for which I present an on-line approximation method. My empirical analysis covering 54,000 videos shows that I can perform this confirmation attack with surprising speed and accuracy.

6.2 Overview

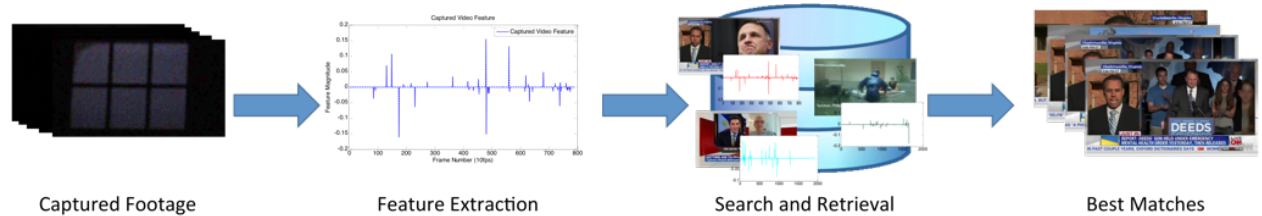


Figure 6.1: The high-level workflow of my approach. Features are extracted from the captured video and then compared with features from reference videos in a database. The reference video with the most similar feature is output as the most probable candidate.

The key insight I leverage is that the observable emanations of a display (e.g., a TV or monitor) during presentation of the viewing content induces a distinctive flicker pattern that can be exploited by an adversary. This pattern is observable in a wide range of scenarios, including videos capturing the window of the room housing the display, videos from cameras pointed at a wall in the room but

not at the TV directly, videos observing the watcher’s face (for example, via a Kinect or similar front-mounted camera), and of course, from video capturing the TV directly. To facilitate my attack, I convert the observed pattern of intensity changes into a suitable feature vector that is amenable to rapid matching of other stored feature vectors within the adversary’s large corpus of processed videos.

In this chapter, I compute the average pixel brightness of each frame in the video, resulting in a mean brightness signal for the video. To capture the sharp changes in brightness, I then use the gradient of the signal as the descriptor for the video. The overall process is illustrated in Figure 6.1. Similar to the captured video, every video in the adversary’s collection is represented by a feature based on the gradient of the brightness signal. Note that while the mean brightness signal of the reference video and the captured videos signal may vary, their gradient-based features share more characteristics in common, and it is those commonalities that are used to identify the content being watched.

6.3 Automated Video Retrieval

My approach (as shown in Figure 6.1) consists of two main parts that comprise a *feature extraction* step from the captured recording and a *video retrieval* step using a precomputed library of features from reference content (i.e., the set of videos for which the adversary wishes to confirm her hypotheses against). The feature extraction stage converts the captured video into a representative encoding that encodes the changes in brightness. This feature is then compared during the video retrieval to the features in the database to identify the content on the victim’s display.

6.3.1 Feature Extraction

Intuitively, in my feature representation I want to preserve the brightness changes of the displayed image. Hence, for each frame t of the video, with M frames, I calculate the average intensity $s(t)$ with $t = 0, \dots, M$, by averaging all the brightness values of all pixels in the image. The sequence s of these average brightness values $s(t)$ with $t = 0, \dots, M$, provides us with a coarse

characterization of the captured video's brightness. An example brightness sequence for a video s_c captured through the window and the corresponding original video s_d is provided in Figure 6.2. Notice that while the variation of the two signals is comparable, the magnitude of the brightness signals $s(t)$ is significantly different.

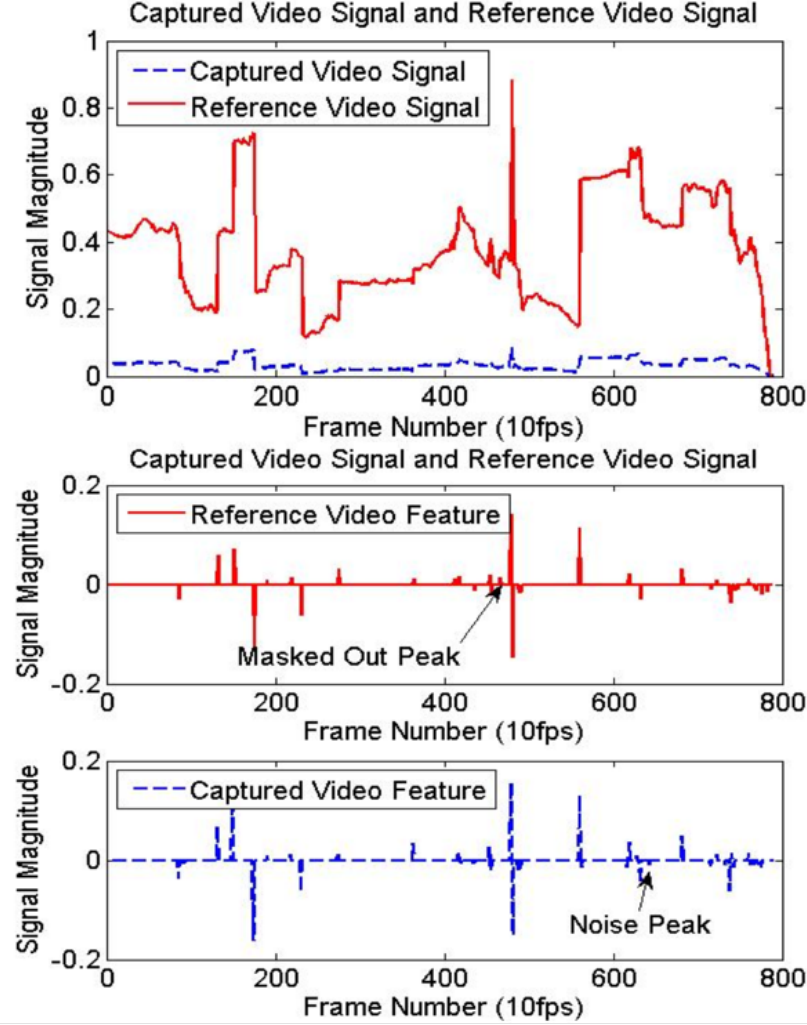


Figure 6.2: The intensity signal (top) and respective features (middle and bottom). For illustrative purposes, the sequences are manually aligned. Noises occur as peaks or masked out peaks in the feature sequence

To achieve comparability of the two signals s_c and s_d I characterize them by their frame-wise intensity gradient over time $ds(t)$. Given the average intensity signals s_c and s_r respectively, the temporal gradient $ds(t)$ can be calculated as $ds(t) = s(t + 1) - s(t)$.

Based on my conjecture that the brightness changes uniquely characterize a video, I convert the temporal gradient ds into a feature f by only preserving its significant ($|ds(t)| > 1$) local maxima and minima

$$f(t) = \begin{cases} ds(t), & \text{if } |ds(t)| > 1 \wedge |ds(t)| > |ds(t-1)| \\ & \wedge |ds(t)| > |ds(t+1)| \\ 0, & \text{else} \end{cases} \quad (6.1)$$

If the video sequence's brightness does not have scene changes, flashes or other sudden changes, the average intensity is nearly constant, leading to zero values in $f(t)$. By contrast, if there is a sudden intensity change (e.g., a drastic scene change or flashes of gun shots) $f(t)$ will capture a "peak" which is either positive or negative, representing a sudden increase or decrease in average intensity. Accordingly, $f(t)$ can be viewed as a composition of peaks. For a captured video, some of the peaks might correspond to noise or noise might mask some peaks in f . Additionally, the magnitude of the peaks might be still scaled by an unknown factor. Example features f_c and f_d for a captured video and the retrieved database video are shown in Figure 6.2 (middle, bottom).

6.3.2 Creating the Reference Library

My video retrieval requires a database of reference videos to retrieve the corresponding video being watched. This database is typically obtained ahead of time by obtaining all content of interest. If only the content for live TV is of interest to the adversary, she can just record all the currently running TV channels. If the adversary is interested in online videos, a database of popular videos (e.g., from YouTube, Netflix, or her home collection) would be helpful. Once all videos of interest are obtained, they are converted to feature vectors using the same feature extraction technique used for the captured sequences (see Section 6.3.1).

6.3.3 Locating the Best Matching Sequences

To identify the best match I use a nearest neighbor search across subsequences because the captured sequence typically only covers a small part of the overall content being watched on the

display. For ease of exposition, I first introduce my similarity metric for the case that both the captured length $length(f_c)$ and the reference video length $length(f)$ are the same and start at the same time. Later, I generalize the metric to account for different lengths and starting points of the captured and the reference videos.

Intuitively, to measure the similarity of the feature vectors for the captured video f_c and a reference video $f_i \in \{f_{ref}\}$, I can examine how many extrema match between the features. The amount of disturbance caused by erroneous noise peaks is represented by

$$E_{noise}(f_i, f_c) = \frac{\sum_{t=1}^L f_c(t)^2 1(f_i(t) == 0)}{\sum_{t=1}^L f_c(t)^2} \quad (6.2)$$

where L is the length of the videos and $1(x)$ is the indicator function, which is one if x is true and zero otherwise. Similarly

$$E_{miss}(f_i, f_c) = \frac{\sum_{t=1}^L f_i(t)^2 1(f_c(t) == 0)}{\sum_{t=1}^L f_i(t)^2} \quad (6.3)$$

measures the energy of missing peaks in the reference sequence. Note that while E_{noise} and E_{miss} characterize the magnitude of difference in the number of peaks, I must also measure the amount of difference in energy of the peaks by characterizing how similar the extrema themselves are. This can be measured as the correlation $Corr(f_i, f_c)$

$$Corr(f_i, f_c) = \frac{\sum_{t=1}^L f_c(t) f_i(t)}{\sqrt{(\sum_{t=1}^L f_c(t)^2)(\sum_{t=1}^L f_i(t)^2)}} \quad (6.4)$$

between the two sequences, which has a value between -1 and 1. In this chapter, I use a similarity metric d that combines E_{noise} , E_{miss} and $Corr(f_i, f_c)$ into a single metric:

$$\begin{aligned} d(f_c, f_i) = & \alpha (E_{noise}(f_i, f_c) + E_{miss}(f_i, f_c)) \\ & + (1 - Corr(f_i, f_c)) \end{aligned} \quad (6.5)$$

with α representing the weighting between the energy of the missing or noise peaks and the correlation between the correct extrema; the latter is necessary when distinguishing features in the case of perfectly agreeing peaks. Given that the magnitudes of the peaks may be different between the captured and reference signals, I rely on the temporal information which is more accurate. As such, I empirically chose $\alpha = 50$ for all my experiments so that the temporal agreement of peaks dominates the metric. It is only when the temporal position of peaks matches perfectly that $Corr(f_i, f_c)$ is used to evaluate their similarity based on magnitude.

Returning to §2.4.1.2, it is important to remind the reader that my metric is based on the gradient of average intensity. Therefore, it captures sharp intensity changes and ignores smooth terms such as ambient light condition, the auto exposure of camera and other gradual changes. Even impulse noise (e.g. turning on/off the room light) only results in a single extra peak in the feature vector and thus has minor impact on the overall result. Other alternatives such as using the correlation directly (e.g., as proposed by Greveler et al. (2012)) fail in my scenario since these approaches are significantly impacted by signal magnitudes which are often heavily distorted. Likewise, the FFT transformation used in sequence matching schemes (Faloutsos et al., 1994; Moon et al., 2002) also fails because the peaks are too sparse for frequency analysis and the localized changes are too subtle to be useful.

In my evaluations that follow, the reference video that best matches under my similarity metric d is reported as the likely content being watched. I note that in practice the temporal position of the extrema may vary by one frame due to encoding and sampling of the original video sequence. Therefore, when determining whether $f_i(t)$ or $f_c(t)$ is non-zero, I consider the adjacent two frames $(t - 1, t + 1)$ in addition to the frame at time t by using the modified indicator function $\tilde{1}(x)$ in Equations (6.2)-(6.5), which is one if none of x or its temporal neighbors is true.

Notice that thus far, the retrieval using the similarity metric d from Equation (6.5) assumes equal length and starting point of the videos. To relax this assumption, for a recording of length $l_c = length(f_c)$ I search all subsequences of length l_c among all database sequences of length greater than or equal to l_c . This has the added benefit that I not only identify what content was

watched, but also what part of the video was watched at the time the recording was taken. The problem, however, is this type of exhaustive search comes with a significant computational burden. In what follows, I discuss how to achieve a more efficient solution in practice.

6.4 Illuminati: Efficient Attacks using Compromising Effusions

To tackle large-scale databases of tens of thousands of videos, I employ a matching algorithm that only needs to search a small fraction of the database. Recall that my similarity metric (Equation 6.5) mainly matches significant intensity changes (peaks in my feature representation) in the captured video with the peaks in the database videos. Next, I leverage the fact that these peaks are only present in a small fraction of the video frames and propose a new *peak-feature* that efficiently characterizes the distribution of peaks. This distribution can then be used to narrow down the search space and speed up the search by an order of magnitude. My proposed algorithm consists of two steps. The first step is the extraction of the features based on only the peaks and the second step uses an efficient index-based search.

6.4.1 Peak-feature Extraction

My proposed peak-feature aims at capturing the distribution of the peaks caused by sudden intensity changes in the video. As shown in Figure 6.3, the peak-feature is computed within a sliding window, of size $w = 512$, over the gradient feature, i.e. the peak-feature is computed from the w consecutive feature values. The value 512 is chosen empirically since my experiments indicate that subsequences shorter than 512 frames (at 10 Hz video frame rate) do not provide enough information for retrieval. To limit sensitivity to peaks caused by noise, all peaks with a magnitude lower than a predefined threshold (30% in my experiments) of the strongest peak's magnitude are omitted. The remaining dominant peaks are assumed to stably represent the gradient feature within the window and are encoded into my proposed peak-feature.

The encoding scheme works as follows. A histogram of the pairwise distances between all pairs of peaks is computed. The histogram uses a bin size of eight, which roughly corresponds

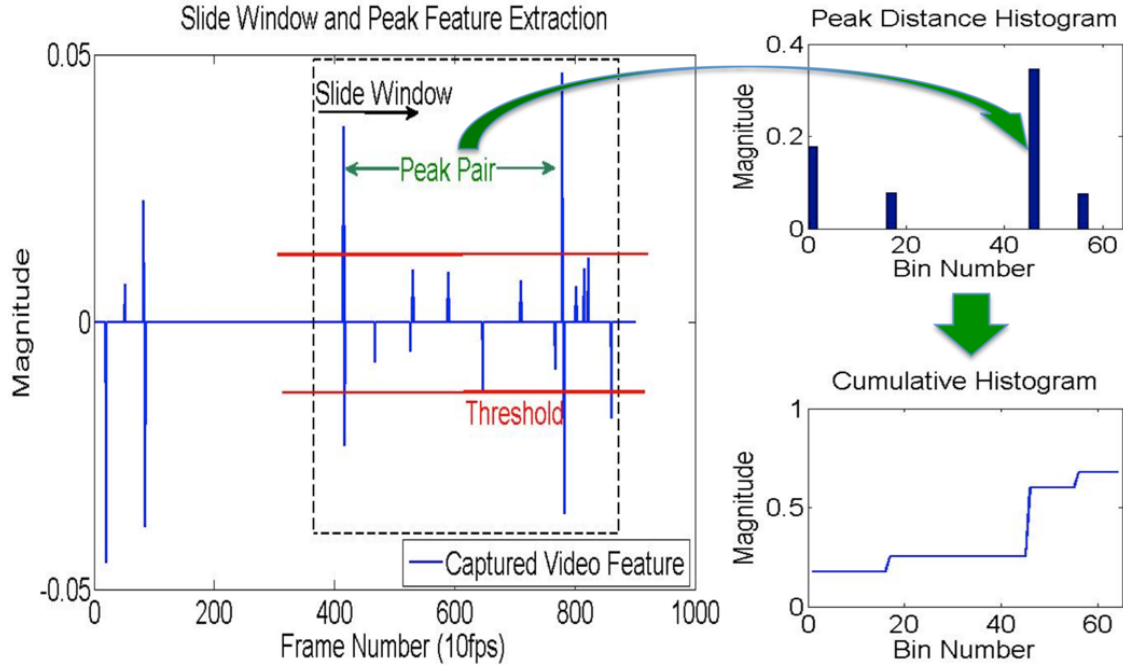


Figure 6.3: Depiction of a sliding window for extracting the peak-descriptor. To suppress noise related peaks, peaks below a predefined threshold are ignored. Effective peaks pairs create a histogram and the cumulation is used as the descriptor of the window.

to a one second distance quantization. The resulting histogram has 64 bins for my window size of 512 frames. Each pair of peaks increases the count in the bin corresponding to their distance (measured by their frame number difference). To model the fact that the stronger peaks are more reliable, the amount of increase is equal to the product of the peaks magnitude. In that way, peaks with larger magnitudes contributes more significantly to the histogram. To ensure comparability between feature windows with different numbers of peaks, I normalize the histogram to sum to one. My peak-feature is the cumulative histogram of the normalized histogram. For the remainder of this chapter, I use this cumulative histogram as it is less prone to the influence of noise caused, for example, by the quantization through the histogram bins.

In summary, my proposed peak-feature is a monotonically increasing 64-dimensional vector with the final element being 1. An example histogram and the corresponding peak-feature is illustrated in Figure 6.3. The distance between two peak-features can be measured by the Euclidean distance of the 64 dimensional vectors. The peak-feature is invariant to the starting point of the window given that it only encodes the pairwise peak distances. When the window slides across the

feature, the peak-feature remains stable as long as there is no peak coming in or going out. For completeness, the exact process is given in Algorithm 1.

An entire video can then be represented as the set of its peak-features, which typically leads to a large set of features describing the video. However, since the peak-feature only depends on the peaks within the window, shifting the window by one frame often results in the same peak-feature (as long as all peaks remain in the window). Empirically, this is the case for about 95% of the peak-features. Accordingly, I represent a video using only its unique peak-features and remove all redundant peak-features from the set of computed peak-features.

Algorithm 1 Extracting peak-feature from window f_{win}

```

1:  $Threshold \leftarrow 0.3$ 
2: for  $i = 1$  to  $N$  do
3:   if  $|f_{win}[i]| < Threshold * \max(|f_{win}|)$  then
4:      $f_{win}[i] \leftarrow 0$ 
5:   end if
6: end for
7: for  $i = 1$  to 64 do
8:    $Histogram[i] \leftarrow 0$ 
9: end for
10: for every 2 peaks  $p_i, p_j$  in  $f_{win}$  do
11:    $Histogram[dist(p_i, p_j)] \leftarrow Histogram[dist(p_i, p_j)] + |p_i p_j|$ 
12: end for
13:  $Histogram \leftarrow Histogram / \text{sum}(Histogram)$ 
14: for  $i = 1$  to 64 do
15:    $PeakFeature[i] = \sum_{k=1}^i Histogram[k]$ 
16: end for
17: return  $PeakFeature$ 

```

6.4.2 Efficient Searching

Next, I detail my proposed efficient search algorithm, which leverages the introduced peak-feature for efficient search. Algorithm 2 provides the pseudo-code for my method and will be detailed below. Given a recording of interest, I first extract the peak-features for the video (see *line* 6). Peak features with a high number of strong peaks are typically very distinguishing, having a

Euclidian norm that is typically larger than peak-features with weaker or fewer peaks. During the matching process, I select the peak-feature with the largest norm first (see *line 7*).

To search the database for a likely match (see *line 9*), I index the peak-features using a data-structure known as K-d tree, which is widely used for search in high-dimensional search spaces (Bentley, 1975). The main idea of the K-d tree is to recursively split the space with hyperplanes, which iteratively refines the possible location of the data point under examination. In my empirical evaluations, the reference library contains 27-million peak-features representing the 54,000 videos. By leveraging a K-d-tree, I can quickly search for all reference videos that are likely matches. Here, a likely matching video has to be within a Euclidean distance of $\delta \leq 0.7$ from the peak-feature of the captured video².

From the likely matches I select the one with the smallest Euclidian distance to the captured video (see *line 10*). For this video my similarity metric from Equation 6.5 is computed. If the similarity is the best observed similarity thus far, this video is retained as the top candidate and its confidence is increased (see *line 16*). Then, the next strongest peak-feature is obtained (see *line 18*) and evaluated in the same manner (see *line 9-16*). If the retrieved video is the same as the previously selected one, the confidence assigned to this potential match increases (see *line 16*). Otherwise the newly found best match replaces the previously selected best video (see *line 12-14*). This process is repeated until the best-matching video remains stable for three consecutive trials.

The algorithm proposed above is an offline approach, which can be extended to operate in an online fashion. For offline retrieval, I have access to all the peak-features at once. Hence, I have the luxury of ranking the features by strength. In contrast, for online operation, the video is streamed. Once a new frame is captured a new peak-feature is computed using the 512 most recent frames. If the newly computed feature is unique for the video, i.e., has not been extracted from the video before, the K-d tree is used to search for likely matches within the reference library. Then the best video (i.e., with the smallest Euclidian distance) is fully evaluated using my proposed

² The value for δ was empirically chosen based on a rudimentary analysis of the resulting accuracy.

Algorithm 2 Efficient searching captured feature f_c

```
1:  $BestScore \leftarrow INF$  // best so far score
2:  $BestId \leftarrow INF$  // database id of best candidate
3:  $ConsecutiveHits \leftarrow 0$  // number of consecutive confirmation of best candidate
4:  $MaxHits \leftarrow 3$ 
5:  $Radius \leftarrow 0.7$ 
6:  $PeakFeature \leftarrow extractPeakFeature(f_c)$ 
7:  $CurFea \leftarrow featureOfStrongestPeak(PeakFeature)$ 
8: while  $exist(CurFea)$  and  $ConsecutiveHits < MaxHits$  do
9:    $RefFea \leftarrow searchKdtree(CurFea, Kdtree, Radius)$ 
10:   $[CurScore, CurId] \leftarrow findMinSMetric(CurFea, RefFea)$ 
11:  if  $CurScore < BestScore$  then
12:     $BestScore \leftarrow CurScore$ 
13:     $BestId \leftarrow CurId$ 
14:     $ConsecutiveHits \leftarrow 0$ 
15:  else
16:     $ConsecutiveHits \leftarrow ConsecutiveHits + 1$ 
17:  end if
18:   $CurFea \leftarrow featureOfNextStrongestPeak(PeakFeature)$ 
19: end while
20: return  $BestId$ 
```

similarity metric from Equation (6.5). If the best video is identical to the previously identified one, its confidence is increased. Otherwise it replaces the current best choice.

On Efficiency: Levering the peak-features and the K-d tree based search reduces the search time on average to less than 10s (2.8 seconds for each K-d tree search) for a database of 54,000 reference videos. The achieved query time is more than an order of magnitude faster than searching exhaustively through the database, which took 188s. The online search can in fact be executed in real time when allowing a latency of 512 frames due to the required temporally preceding information for the peak-feature computation.

6.5 Evaluation

For my empirical evaluation, I collected a large collection of reference videos spanning a wide variety of content. My reference library contains 10,000 blockbuster movies of at least an hour in length, 24,000 news clips ranging from 5 min to 20 min each, 10,000 music videos ranging from 2

min to 7 min each, and 10,000 TV-shows ranging from 5 min to 20 min each. In total, the library indexes over 18,800 hours of video. All features and peak-features from the library are precomputed by leveraging my proposed methods from Sections 6.3 and 6.4. For my experimental evaluation I randomly selected 62 sequences as my test set of videos.

For the first set of evaluations the test videos were played on a 24 inch screen with no additional room lighting turned on. I then capture the reflection of the screen emanation from a white wall at a distance of three meters from the screen. To capture the video, a Logitech HD Pro Webcam C920 and a 60D canon DSLR were used. I run the experiment in a home environment as well as in a lab environment. The setting of my experiment is illustrated in Figure 6.4. These captured videos were then used to execute my attack. For these evaluations I assess the success of the attack with respect to the duration of the captured video and the size of the reference library.

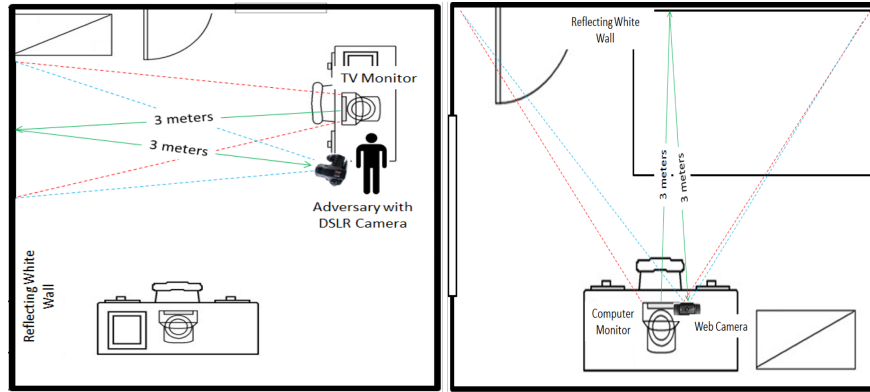


Figure 6.4: Lab environment (left) and home environment setting (right)

Lights Off

First I evaluate the success rate of my method using a room with the lights off, as commonly occurs when watching TV. A success is the correct identification of the video being watched. I do not leverage any knowledge about the video being played nor do any of my experiments use any knowledge of the scene or the capture distances. The time at which the adversary starts capturing the emanations from the display is chosen at random.

Capture Length	60s	90s	120s	180s	240s	270s
Success Rate	39%	49%	54%	70%	85%	94 %

Table 6.1: Retrieval success rate with random start point.

For the 62 test sequences I analyzed segments from 60 to 270 seconds long. These segments are processed by the feature and peak-feature extraction procedures. The resulting features and peak-features are then used to infer the best match among the reference library. The experiment is repeated 100 times for each of the different segment lengths, each time choosing a random starting position. Table 6.1 shows the resulting average success rate over all starting positions. As expected, the longer the captured sequence, the higher the attack's success rate. The results shows that the success rate increases from 39% for a 60 second segment to 94% for 270 seconds, and has nearly a 50% success rate using only 90 seconds of captured emanations. A more detailed analysis of the data reveals that in the limit, the success rate is 100% for each video as subsequences within these videos can always be uniquely identified.

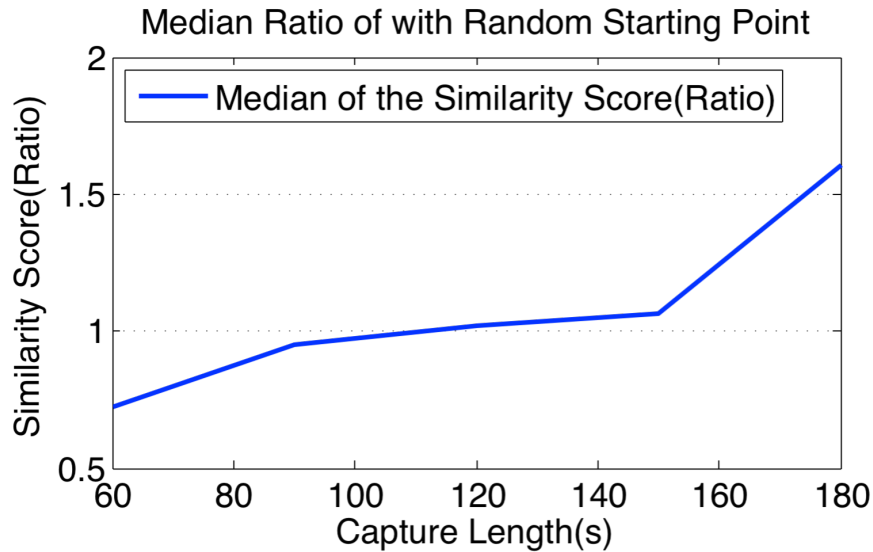


Figure 6.5: The median ratio within the dataset of 54,000 references.

To better quantify the robustness of my approach, I evaluate the ratio in similarity between the video sequence returned as the best match and the true positive. If the ratio is larger than one, that implies the correct video will always be identified. The higher that ratio, the more distinct the

retrieval result. Obviously, the outcome also depends on the contents of the reference library itself. The experimental results of the ratio evaluation are shown in Figure 6.5. The median similarity score ratio rises above one (successful retrieval) between 100 and 120 seconds. For longer sequences, it monotonically increases with increasing segment length.

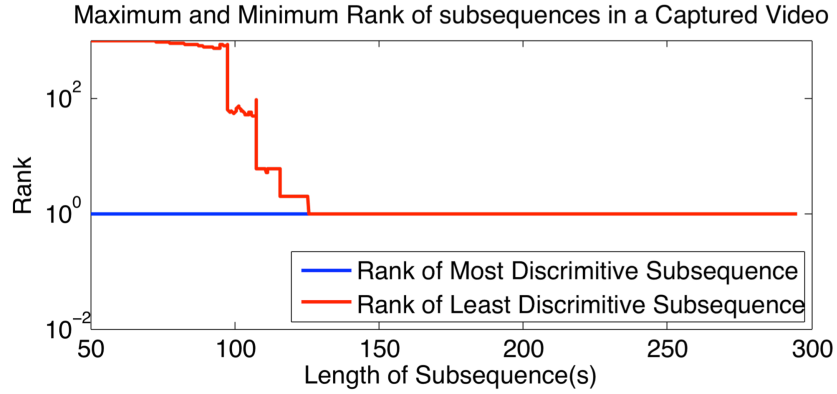


Figure 6.6: Rank of correct video in the best case (blue) and worst case (red).

Beyond the average success rate and robustness, it is also important to understand the best and worst case results. The worst case is especially useful since it provides a measurement for an attacker of how much video is needed to reliably achieve a successful attack. To measure these boundaries I evaluate the retrieval success rates for all possible sub-sequences longer than 10 seconds and all possible starting points for my 62 test sequences. For each of these tests I then rank the retrieved videos by their similarity scores and report the rank of the ground-truth video. If the corresponding video is ranked first the retrieval was successful, otherwise it was not.

Figure 6.6 shows the rank of the corresponding video with respect to the captured video's length for one of my test videos. The results for the other videos are comparable. In the best case, the corresponding reference video is always ranked first, which means if the attacker is lucky enough, she will be able to retrieve the correct video even if she only captures 10 seconds of video. The results also shows that any captured segment longer than 120 seconds within this particular video can always be successfully retrieved.

Next, I summarize the results on a per-video basis by assigning a video its worst segment's similarity ranking, i.e., its worst possible ranking obtained by any of the corresponding video for

Illumination settings	SNR	Segment Length
Normal brightness level room light off	70	180s
50% brightness level room light off	33	270s
Normal brightness level room light on	15	300s

Table 6.2: Worst case capture length with different illumination settings.

any of its segments. This captures the lower bound of the attacks performance for each of my 62 test videos. The results are shown in Figure 6.10. Expectedly, the variation is the largest for the shortest segments of less than 100 seconds and converges to one with capture length longer than 240 seconds.

6.5.1 Lights On

The illumination of a scene (e.g., both room and natural light) contribute significantly to the amount of light entering the camera, which in turn influences the brightness level of the captured video. Obviously, screens with lower brightness naturally reduce the light emanation. Therefore, I evaluate the influence of the illumination on the performance of my proposed attack. In this experiment I use a 24 inch screen, and the attacker’s camera captures the reflection of the screen of a white wall, which is three meters away from the screen. The camera used in the attack is a Canon Rebel T4i DSLR. I captured five videos in each of three different illumination settings: 1) normal screen brightness with room light off, 2) 50% reduced screen brightness level with room light off, 3) normal screen brightness with the room light on. The obtained retrieval results are shown in Table 6.2 and Figure 6.7.

The results indicate that higher screen brightness levels make the retrieval slightly more successful as it takes shorter segment lengths for successful retrieval and the similarity ratio is higher. However, the influence of the screen brightness seems marginal. It can also be seen from Table 6.2 and Figure 6.7 that even with the room light on, my attack is successful with moderate segment length, which I attribute that to my robust similarity metric. The only effect that both the lower screen brightness and the active room light have is that it mandates that longer segments are necessary for successful retrieval in the worst case. This is expected, as in both cases,

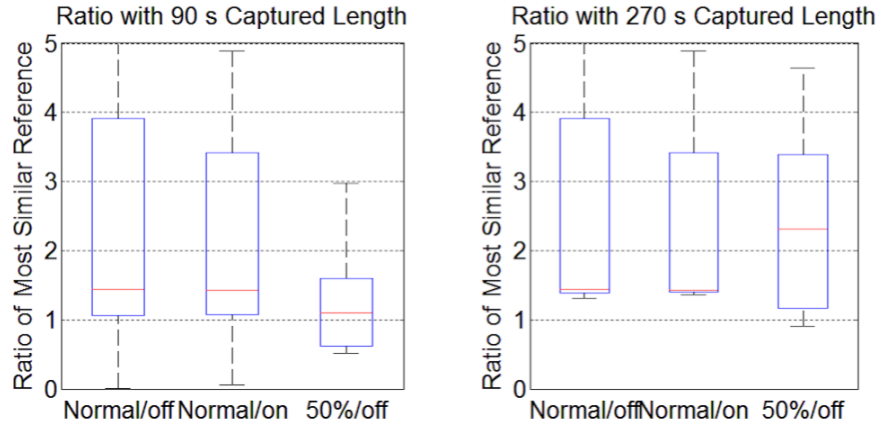


Figure 6.7: Ratio to second-best under different illumination conditions.

smaller, less significant brightness changes are not detectable anymore. Accordingly, there are fewer distinguishing elements I can use. In the case of the active room light, I only failed once when retrieving a video based on a segment that was 270 seconds, but succeeded with a 300 second segment. It is worth noting that in the case of an active room light, a human observer is not able to perceive the resulting subtle intensity changes on the wall.

6.5.2 Impact of Screen Size

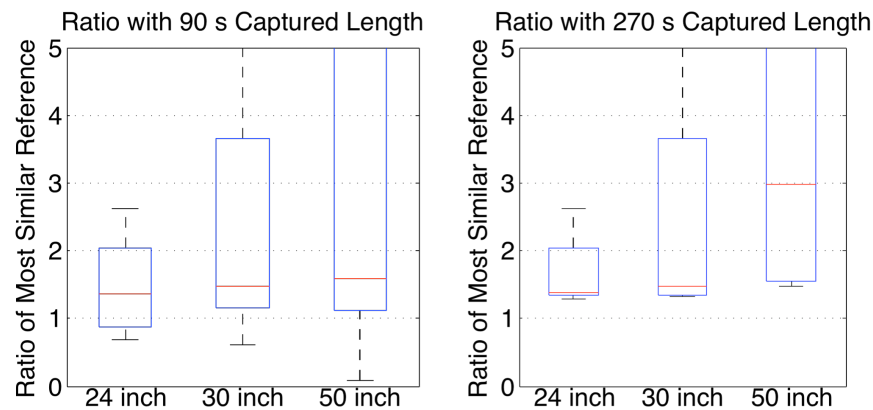


Figure 6.8: Boxplot of the second-best ratio w.r.t. different screen sizes.

The amount of light captured by a camera not only depends on the screen's illumination setting but also on the actual screen size as it influences the amount of light emitted into the environment. Generally, bigger screens emanate more light, which leads to higher quality video capture. To

evaluate the impact of screen size, I performed an experiment in which I used differently sized LCD displays. In particular, I used displays with 24 inch, 30 inch, and 50 inch screen sizes. I again use a Canon Rebel T4i DSLR to capture the video of the back wall, which is 3 m away from the screen. For each screen size I capture five videos. The resulting required worst case segment lengths for successful retrieval are shown in Table 6.3 while Figure 6.8 shows their distribution. The SNR is lower because the experiment was performed in a different room with a lot of light-absorbing materials.

Screen Size	SNR	Worst Case Length
24 inch	5	270s
30 inch	48	180s
50 inch	109	180s

Table 6.3: Worst case capture length with different screen size.

Expectedly, the larger screen size supports better retrieval for shorter segments. The shorter segments that fail on the 24 inch screen can often be successfully retrieved with the 30 and the 50 inch screens. The similarity ratio is higher on larger screens leading to more robust identification.

6.5.3 Impact of Reference Library Size

The retrieval results are influenced by the distribution of the videos within the database and the size of the database. To characterize the change in behavior I compute the worst case ranking for two reference libraries consisting of 1000 and 4000 videos respectively. The results are shown in Figure 6.9. As expected, it can be seen that the larger the database, the longer the segments have to be to guarantee a successful retrieval. However, the increase in segment length with respect to the increase in database size is moderate. For example, for an increase in database size from 4,000 to 54,000 videos (13.5x), the segment length only increases by 20% (from approximately 200 seconds to 240 seconds). I predict that this increase will decline even more for larger databases as the probability of two identical video segments appearing in different videos exponentially decreases with the length of the segment.

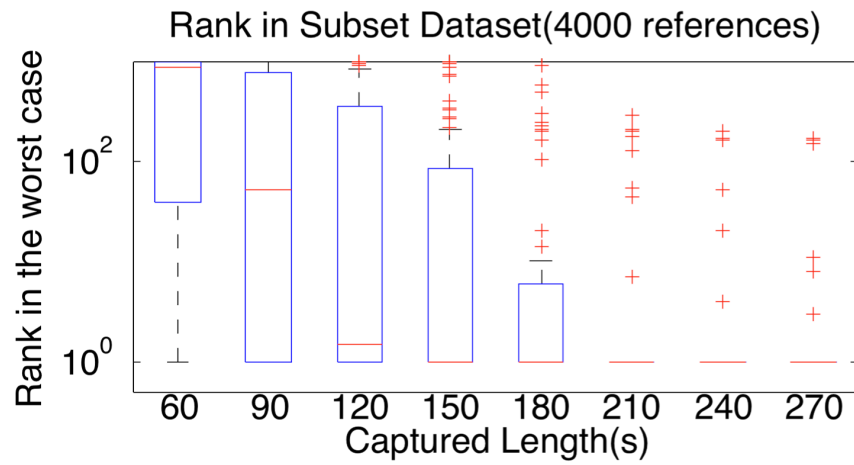
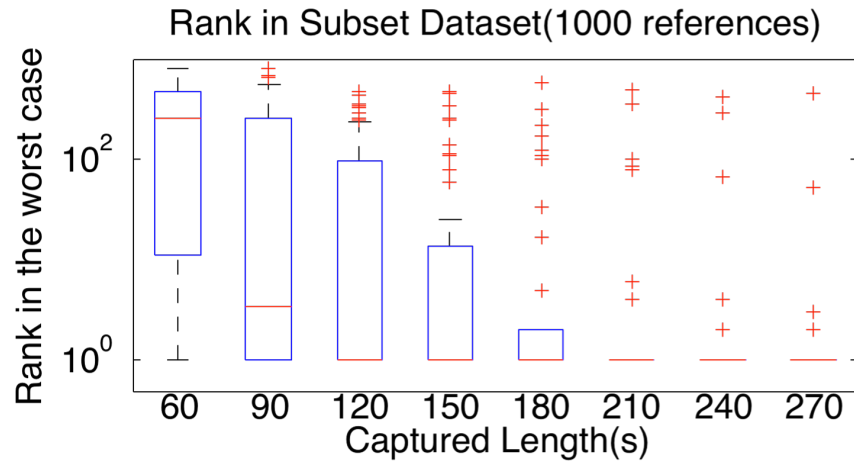


Figure 6.9: Rank of correct video among libraries of size 1,000 and 4,000.

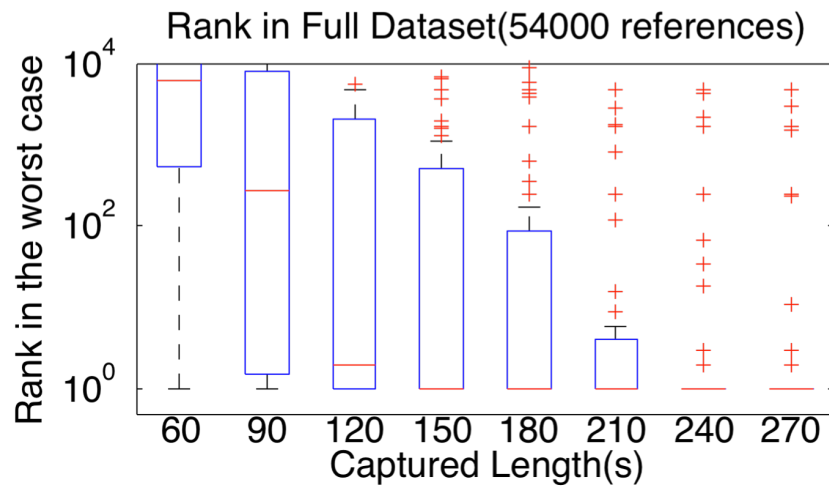


Figure 6.10: Rank of correct video (among 54,000 videos).

6.5.4 As Seen From Outdoors

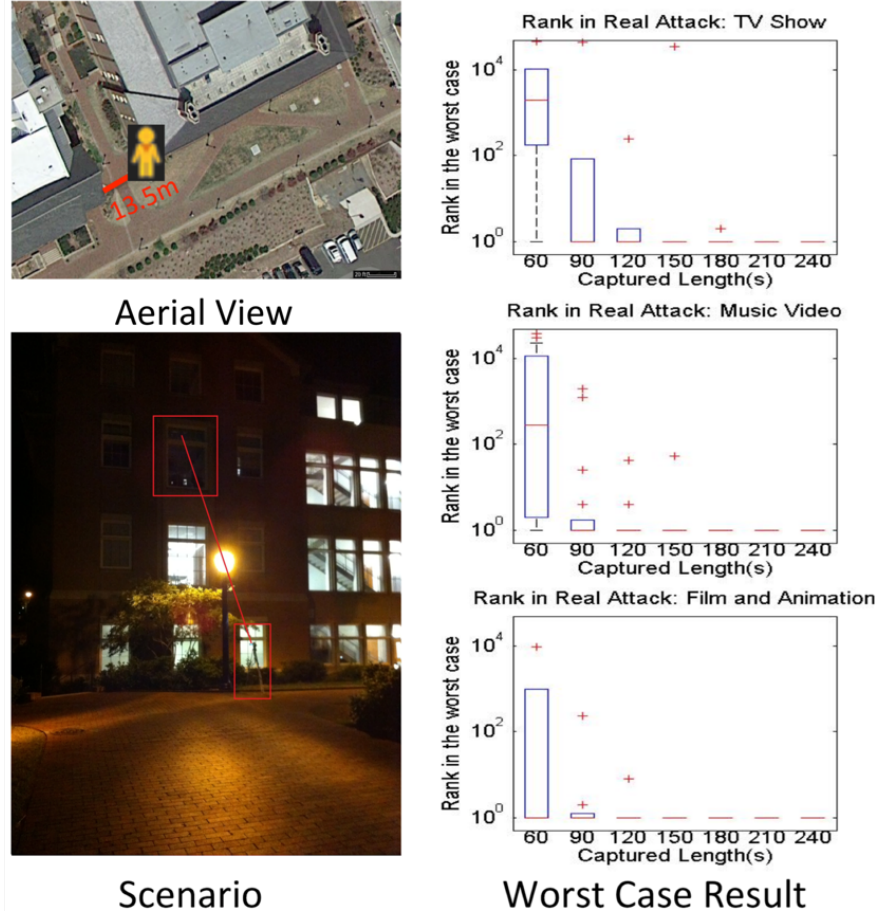


Figure 6.11: TV reflection in the room is captured from a distance of 13.5 meters (left). The worst case results (right) are illustrated for different types of videos: TV shows, music and film from top to bottom. All segments longer than 180s were successfully retrieved.

To further demonstrate the practicality of my proposed attack, I tested its effectiveness from outdoors. I captured the emanations seen on an outside window of a room with a TV showing 60 of my test sequences. In this scenario, the attacker was positioned on the sidewalk observing the third floor office window of the room with the TV (see Figure 6.11). The TV emanations reflected off the beige ceiling of the room and towards the window which was situated orthogonal to the TV. The TV is 13.5 meters away from the adversary. For completeness, I evaluated my approach using videos from varying categories of media that include TV shows, music videos and films. 20 samples of each video type were captured. Figure 6.11 (right) shows the worst case result with respect to

different subsequences. The results indicate similar success across all videos tested, and in all cases, I was able to perform the confirmation attack.

To gauge the robustness of my approach, I further experimented with recordings captured at much further distances. In this case, the attacker was positioned on the sidewalk 70.9 meters from the building; the TV was playing in the same third-floor room as in the previous experiment. TV emanations were captured from the ceiling reflection with the same Canon Camrecorder. 20 sequences randomly selected from different categories are tested. The proposed approach successfully retrieved 18 sequences out of them within 5 minutes. The experimental setting and results are depicted in Figure 6.12. The results are compared with that of direct view and 13.5 meter reflection (Figure 6.12 bottom). In the worst case, the sequence can usually be retrieved within 100 seconds at 13.5 meters away, compared to 190 seconds, on average, from 70 meters away.

6.6 Mitigations

The simplest mitigation is to cover the windows of the room with blackout curtains to effectively avoid the leakage of the light to the outside. To gauge the effectiveness of such a defense I performed a rudimentary experiment with vinyl blinds and curtains (see Figure 6.13)³. The setup was the same as for the attack carried out at 13.5 meters outdoors, except for the use of shades. In this experiment, only two samples were tested in each case. For the case of vinyl blinds and a standard beige curtain with brown stripes, I were still able to determine 3 of the 4 videos being watched after capturing 270s worth of footage. The other video failed to be recovered even after 5 mins. I was unable to confirm any of the watched content when thicker, room darkening, (black) curtains were used.

If the use of curtains is not desired the screen brightness could be lowered to increase the SNR of any captured video. My experimental evaluation demonstrated though that this has only a limited effect on thwarting the attack. My experiments show that retrieval will still be possible as long as the brightness change is perceptible. Although this strategy would not prevent the attack altogether, lowering the screen brightness will increase the burden on the attacker as longer observations

³ The brighter pattern in the middle picture is caused by a reflection on the vinyl blinds from an outside street lamp.

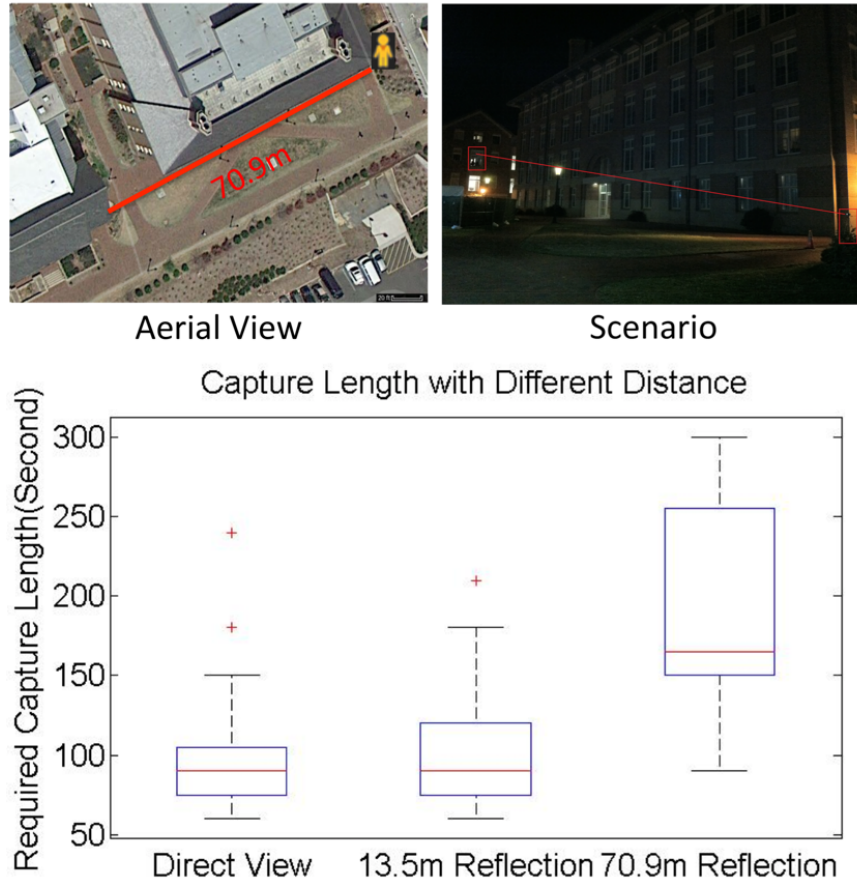


Figure 6.12: TV reflection in the room is captured from a distance of 70.9 meters (top). The camera and the window are labelled in red (top right). The required capture length is compared with direct view and 13.5 meter reflection (bottom). It takes longer for successful retrieval with longer distance.

would be required to successfully carry out the attack. Similarly, the burden on the attacker can be increased if a bright room light is used as that would increase the noise level in the captured signal.

Another defensive strategy may be to install a flood light next to any window of the room so as to effectively blind a camera that tries to observe the diffusions through the window. Doing so would prevent the camera from capturing the subtle brightness changes required to successfully execute the attack. That said, a motivated attacker could overcome this defense by using sophisticated high dynamic range image cameras, which can capture a large dynamic range of light intensities. Alternatively, my attack could be mitigated by installing an adaptive lighting system, which measures the emitted light and counters any brightness change of the emitted light. Doing so would help maintain a constant amount of light emission and would not reveal the brightness change information

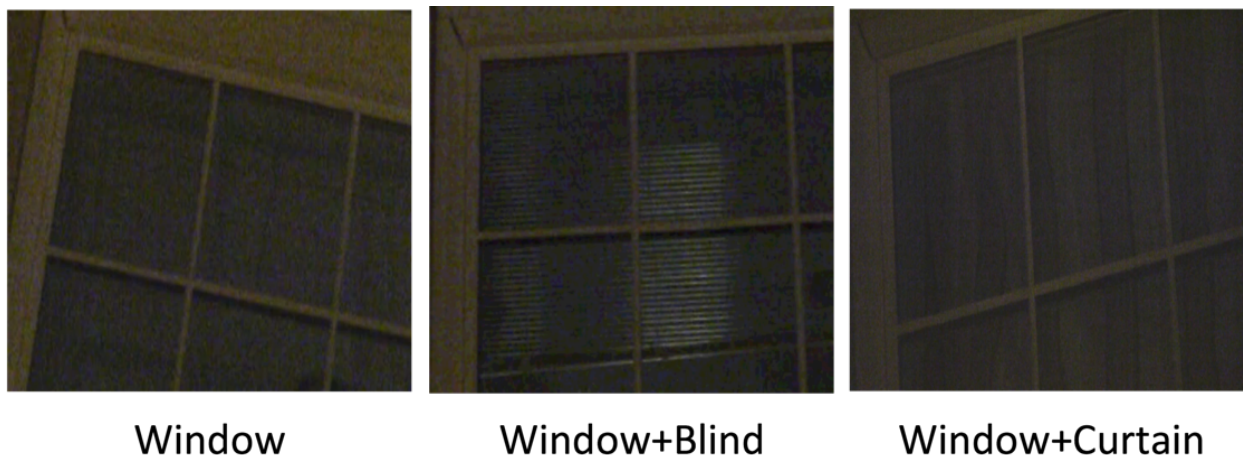


Figure 6.13: Captured image directly from window (left), through vinyl blinds (middle) and through a curtain (right).

to an outside observer. Obviously, these defenses would not be popular in densely populated areas as the outdoor light emissions would likely not be appreciated by neighbors.

6.7 Conclusion and Future Work

In this chapter, I propose a novel method to identify the video content shown on a victim's screen using recordings collected in a number of practical scenarios (e.g., observations of light effusions through the windows or off the walls) and at great distances (e.g., 70m away). My attack shows reliable identification of the content being watched in a wide range of evaluated scenarios. The robustness of the attack is due to a novel application of unique feature sets, a well suited similarity metric, and the development of efficient indexing structures for performing rapid matches in near real-time. My empirical results show that I can successfully confirm hypotheses while capturing short recordings (typically less than 4 minutes long) of the changes in brightness from the victim's display.

Beside TV-content retrieval, my proposed method can be easily applied to detecting co-occurrence in a dataset. Specifically, I can take advantage of my preliminary work which shows that the light emitted by viewing devices or any other modulated or random pattern light source (e.g., flashes of cameras) induces a distinctive flicker pattern that can be used to find linkages among

videos. In this way, I can identify the co-occurrence of two videos even if their viewing perspective have no spatial overlap.

To see why the use of illumination changes offers a powerful capability for link-analysis, consider the example shown in Figure 6.14. The light emitted from a screen is used to infer connections among videos from cameras pointing in opposite directions. The first might be video captured by an on-body police camera, while the other might come from a bystander in the same vicinity, but with her camera turned away from the police officer and instead recording the actions of others nearby. As preliminary work, I explored two capabilities: i) given the video sequence from camera 1, find the best matching sequence from a reference library (that includes video from camera 2), and ii) identify the relationship between the two videos without any *a priori* knowledge of video from either camera. In both cases, the temporal information inferred from intensity changes was successfully used to match the footage from camera 1 and camera 2 among a larger database of 54000 videos. More importantly, once a positive match is found, their linkage can be used to provide even more information for forensic investigations. For instance, known information from one of the videos (e.g., GPS location, captions or other visual text) could be used to automatically populate labels of other matching video(s). For example, given a query video with ashing lights, co-occurrence matches from videos from nearby surveillance or witnesses' cameras that capture similar intensity changes can be linked together.

The preliminary experiment above was conducted in controlled lab settings. In practice, my ability of discovering co-occurrence would be affected by camera movement and surrounding light that is only captured by one camera, which add strong noise peaks in the extracted temporal derivative feature. Therefore, my original approach may need to be extended for the robustness against these factors in order to be effective for co-occurrence detection in the real world. I leave these discussions to future work.

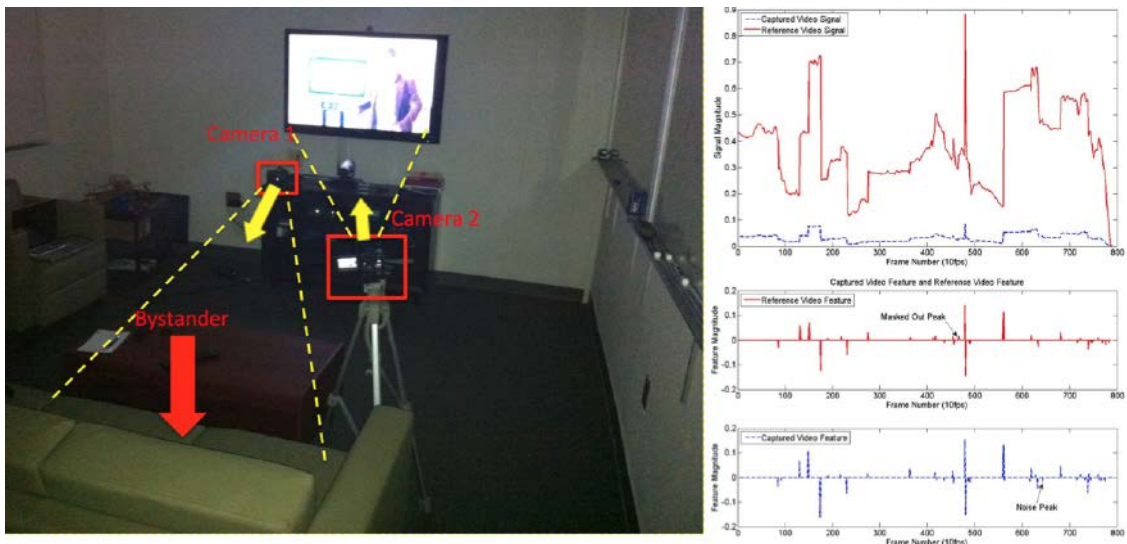


Figure 6.14: (Left): Cameras facing opposite directions; (Right): Encoded intensity signals.

CHAPTER 7: DETECTING COPIES FROM LARGE VIDEO COLLECTIONS

7.1 Introduction

Love it or loathe it, the spread of “pirated” media is here to stay due to its significant economic gain for the pirates. While hard to quantify exactly, the significant economic impact of video piracy can be observed in recent events. The takedown of Megaupload alone has led to 6.5 – 8.5% increase in annual revenue, over \$20 million, for three major motion picture studios (Danaher and Smith, 2014). More recently, the Annual State of Application Security Report from Arxan Technologies (Arxan Technologies, 2015) revealed \$91.6 billion worth of movies and TV series changed hands via illegal distribution networks in 2014. In this chapter, I address the detection of these pirated movies. To comply with the DMCA’s safe harbor protections and stem the tide against such lawsuits, the popular user-generated content (UGC) platforms have turned to automated copyright violation detection. The most popular method is the proprietary Content ID algorithm (YouTube), which inspects daily over 400 years worth of videos.

While these techniques (such as Content ID) have been successful, they have been widely criticized for false detections (Bartholomew, 2014), particularly impacting independent content creators¹. Improving the accuracy of such systems has garnered much attention from both academia and industry given that it remains an important social problem and immense business opportunity. In 2008 the U.S. National Institute of Standards and Technology (NIST) included a separate challenge on video copy detection in their annual TREC Video Retrieval Evaluation (TRECVID) within the IACC dataset (developed in 2006, with over 200 hours of video) (Smeaton et al., 2006). Another challenge dataset was the MUSCLE-VCD-2007 dataset (with over 100 hours of

¹ See also P. Tassi, “*The Injustice of the YouTube Content ID Crackdown Reveals Google’s Dark Side*” Forbes Magazine, 2013.

video) (Law-To et al., 2007). These datasets used synthetic video-based transformations to test the performance of different content detection approaches. After just four years, the TRECVID challenge was prematurely terminated, claiming that near-duplicate video detection was a solved problem. However, in 2014 Jiang *et al.* (Jiang et al., 2014) showed that the current state of the art — which showed near-perfect results on the simulated benchmarks — are far from satisfactory in detecting complex real-world copies. To highlight the problem, Jiang *et al.* (Jiang et al., 2014) released a new dataset (called VCDB) that contains pirated videos available on YouTube and MetaCafe. Their preliminary evaluations suggest that the transformations observed in the real world are very different from the synthetic transformations considered by the academic community to date. For instance, the most widely studied transformation in the TRECVID evaluations, “picture in picture,” is rarely seen in real cases, while the more commonly observed transformations in online pirated videos are far more complex than the synthetic ones tested in past evaluations. As a result, the techniques that appear to be robust in simulated benchmarks fail miserably in the wild. The work of Jiang *et al.* (Jiang et al., 2014) demonstrated that the assumptions made in the NIST challenges were far too naïve and did not generalize well to what real-world adversaries would do. Common transformations observed in the real world are cam recording (filming the screen), brightness and color adjustments, flipping (mirroring on vertical axis), and scaling. I show examples of these in Figure 7.1 . The VCDB dataset itself is a dataset that includes samples which are not detected by current commercial systems, in particular ContentID. Although the dataset is biased, it contains exactly the pirated videos of interest for improving the state of the art. Hence, I use this dataset to evaluate my method for its potential for improving copyright violation (see Section 7.3).

I leverage the observations of Jiang *et al.* (Jiang et al., 2014) regarding temporal consistency for practical content-based copy detection. They observed that in the real world there is a limited range within which the pirates modify the temporal information, *i.e.* frame rate, of a copied video. Outside of this range, it is empirically difficult to thwart copy detection systems without significantly degrading the user’s viewing experience, which would clearly be contrary to the pirates’ goals. I leverage this key observation along with the robustness of the mean image brightness gradient



Figure 7.1: Copy-reference pairs in the VCDB dataset

signal (Xu et al., 2014a) to propose a novel approach based on exploiting *temporal consistency*. In contrast to the existing state of the art, my approach shows significantly improved simultaneous robustness against both spatial transformations and commonly used temporal transformations.

7.2 Approach

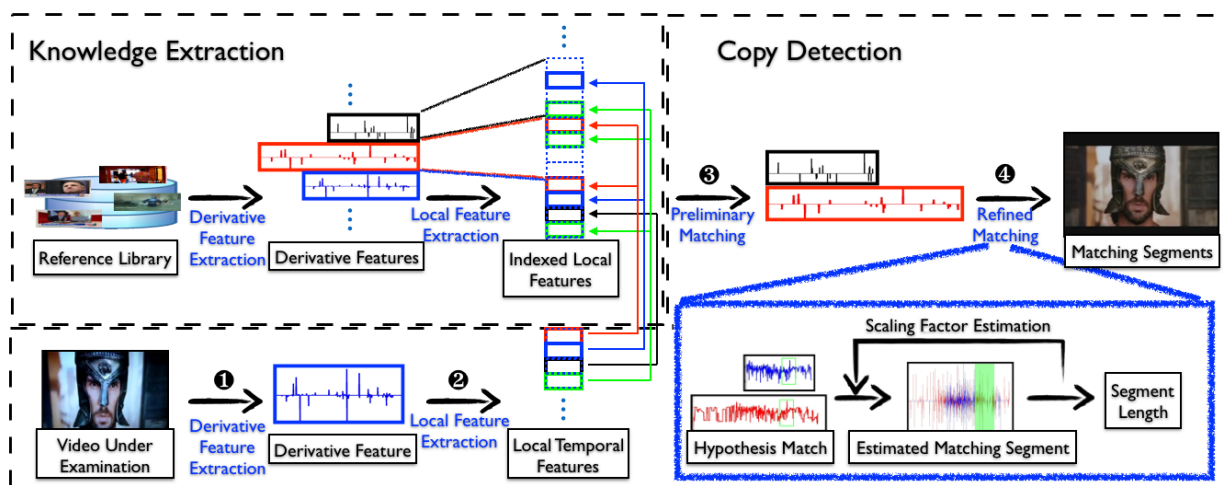


Figure 7.2: Overview of the proposed approach

My proposed method (Figure 7.2) consists of two main components: a knowledge extraction stage and a video sub-sequence matching (“copy detection”) stage. In the knowledge extraction

stage, I first pre-process the reference library by extracting derivative features for all its videos (*i.e.* , computing their “fingerprints” in stage ❶). As noted by Xu *et al.* (Xu et al., 2014a), these derivative features encode the temporal position of illumination changes (peaks) in the videos, which can be leveraged to find linkages to a collection of videos.

To build an indexing structure that is robust against temporal transformations, local temporal features covering a short temporal windows around a peak are extracted from the derivative features of each video (stage ❷)². They encode the gradient signal profile of significant illumination changes in a manner that is robust to temporal transformations. A K-d tree structure is built upon the local features of all reference videos.

The detection stage is used to determine if a video contains infringing material. It also computes the uploaded video’s derivative and local features. The local features are then used as a first approximation (stage ❸) to retrieve reference videos with similar local intensity profiles. Each retrieved local feature provides a set of potential candidates for matching reference videos. The collection of possible matches is then considered for further scrutiny using full video information (stage ❹). The comparison to the full video, or large segments thereof, requires a known temporal transformation. This temporal transformation is provided by a scale factor estimation based on a classifier (SVM) based subsequence alignment. The sequence alignment is then iteratively refined. The aligned video subsequences are finally classified into infringing and genuine videos.

7.2.1 Derivative Feature Extraction

My derivative features are computed as the temporal gradient of the video frames’ average intensity signal s_t . The temporal gradient is calculated as $\Delta s(t) = s(t + 1) - s(t)$. Based on the observation of our previous work (Xu et al., 2014a) that brightness changes uniquely characterize a video, I convert the temporal gradient $\Delta s(t)$ into the derivative feature f by only preserving its

² For ease of readability, I henceforth use the term “local temporal feature” and “local feature” interchangeably as my methods does not use spatial features.

significant local extrema (*i.e.* $|\Delta s(t)| > 1$), given by

$$f(t) = \begin{cases} \Delta s(t), & |\Delta s(t)| > 1 \wedge |\Delta s(t)| > \min\{|\Delta s(t-1)|, |\Delta s(t+1)|\} \\ 0, & \text{else} \end{cases} \quad (7.1)$$

I extend the single feature per frame of Xu *et al.* (Xu et al., 2014a) to a hierarchical representation also computing the temporal feature on a grid of $n \times n$ image tiles, which span the entire image ($n = 3$ in all experiments). This results in a 10-dimensional feature for each frame.

These features are then used to obtain an alignment between a potential uploaded copy (query) and the potential original (database) videos. I pose this alignment problem as a classification problem of the element-wise difference of the feature of a query and a database video. This yields a 10-dimensional similarity score vector. This score vector mainly encodes the difference between temporal positions of changes in illumination between the query and the database video. To consider the intrinsic properties of the uploaded video’s segment, I concatenate the similarity score vector with the length of the uploaded video and the peak rate to form a 12 dimension vector. To classify for alignment, I then train an SVM on these vectors, using a training set of 20% of my manually labeled sequence-pairs as positive samples and an equivalent number of randomly selected negative samples. The classifier outputs a score that can be interpreted as the likelihood that a particular sequence pair is a copy. If the score is higher than a threshold κ , I declare the sequence pair to be a near-duplicate.

7.2.2 Local Feature Extraction

For efficient retrieval, local features are computed from the derivative feature values using a sliding window of $t = 5$ seconds at a 10 Hz video sampling rate (as shown on the left of Figure 7.3). I focus on the sliding window position where there is a peak at the center of the window. Moreover, I only consider the sliding windows where the magnitude of the center peak is larger than at least 60% of other peaks in the window. This choice is guided by the intuition that high

peaks — representing strong changes in illumination — are less likely to be noise and are therefore more likely to be preserved during the creation of a pirated version of the original video.

To achieve robustness to temporal transformations, each peak in the window of a reference video casts votes in its corresponding temporal neighborhood of the query video. The votes for the temporal positions follow a Gaussian distribution. Given the fact that the temporal uncertainty quadratically increases with its distance to the window center, the standard deviation parameter of the Gaussian can be computed based on the expected range of temporal transformations (see left-middle in Figure 7.3 for an example). The resulting vote vector for the window defines its local feature when is normalized to L_2 -norm of one. Next, I discuss on how to index these local features for efficient retrieval.

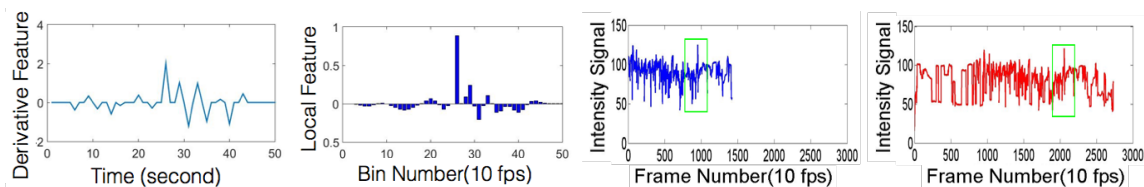


Figure 7.3: Left : sliding window on derivative feature. Left middle: local feature with bounded uncertainty w.r.t. temporal adjustments. Right middle: local feature in uploaded video and (right) its closest neighbor in the reference set.

7.2.3 Indexing and Retrieval

I use a K-d tree (Bentley, 1975) to efficiently index the local features. I note that the VCDB dataset contains over 27 hours of video, wherein I identified 17,438 local features. Given a new video, I extract local features, after which the N strongest features of every five-minutes-long sub-sequence are used to locate similar local features among all reference videos using the K-d tree structure.

For each of these N features, I retrieve the nearest α fraction of local features from K-d tree. Figure 7.3 shows an example of a sample video’s local feature and its closest neighbor. I empirically found that this fraction only needs to grow marginally (sub-linearly) with database size (see Section 7.3). This produces a list of potentially infringing sub-sequences of the query and their

correspondences in the database. However, given the limited temporal context of the local features, these correspondences almost certainly contain false positives.

7.2.4 Detecting Copied sub-sequences

To minimize false positives, I use the frame correspondences as seeds for a more thorough duplicate sub-sequence detection step. This step determines the sub-sequence that has most likely been copied. To correctly identify these copied sub-sequences, I need to determine both their temporal positions and the applied temporal scales.

Sub-sequence estimation: To achieve this goal, I begin my sub-sequence estimation by assuming a scaling factor of one. Then I align the reference video and the uploaded video by matching the suggested frame correspondence. Next, I exhaustively evaluate all sub-sequences containing the frame correspondence with length less than $\Theta = 60$ seconds. The choice of Θ is guided by the fact that 60 seconds is already enough for the trained classifier to determine if the sub-sequence pair is a copy. The actual copy sub-sequence is then determined as the one with the largest SVM score.

Temporal scale estimation Once a sub-sequence is identified as containing potentially infringing content, I refine the scale estimate for that sub-sequence. Given that, in practice, the observed range of temporal scales is limited (see §2.5), for practical reasons I explore scales from 0.8 to 1.2 quantized into 100 steps. The scale that yields the highest similarity between the sub-sequence from the uploaded and the reference video is selected as the scale of the sub-sequence. In order to normalize temporal uncertainties of the peaks, I change the sub-sequence’s temporal domain to its logarithmically transformed time axis with the center peak at the origin. In this logarithmic domain, the uncertainty caused by the scale estimation error becomes uniform across the sub-sequence, and the impact of quantization error is mitigated when estimating the scaling factor. I use this modified scoring scheme to determine the best scale estimation for the identified overlapping sub-sequence (see Section 7.2.4). I then iterate over subsequence estimation and temporal scale estimation. In practice, I found this process to converge within three iterations. An example of a correctly scaled and matched sub-sequence for a pirated video is shown in Figure 7.4. As a final optimization, I only

consider sub-sequences flagged as potential copies if their score from SVM classifier is higher than Φ . The threshold Φ can be tuned to balance precision and recall rate.

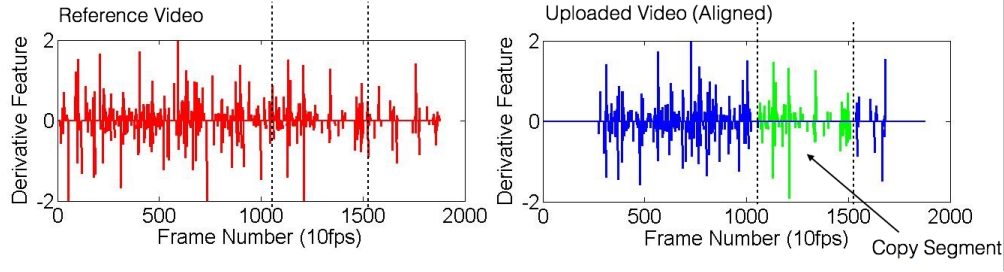


Figure 7.4: Final result of comparing the uploaded video and the selected reference video. Green: matched segment; Blue: Unmatch segment; Red: reference sequence.

7.3 Evaluation

In order to address the piracy of longer content such as movies, TV-series, sports events, podcasts, etc., I evaluated the performance of my approach on videos in the VCDB dataset that are longer than 30 seconds in length. To provide a comparison with the state of the art, I use the same evaluation methodology used in the VCDB benchmark, whereby performance is measured using the standardized metrics of precision and recall. Specifically, the segment-level precision (SP) and recall (SR) in these benchmarks are:

$$SP = \frac{|correctly\ retrieved\ segments|}{|all\ retrieved\ segments|}, \quad SR = \frac{|correctly\ retrieved\ segments|}{|ground - truth\ copy\ segments|}$$

To plot the precision-recall curves, I vary the threshold of the minimum copy-segment length Φ . I set $\alpha = 1.7\%$ and $N = 8$ for all my experiments.

To directly compare my results with the current state of the art, I implemented the best-performing approach on the VCDB benchmark, Tan *et al.* (Tan et al., 2009)’s temporal network approach (TNP), which achieves benchmark performance of 60.92% mean average precision. My implementation reproduced the results reported by Jiang *et al.* (Jiang et al., 2014). In contrast, my method reaches a mean average precision of 69.72% and clearly outperforms the TNP approach. Figure 7.5 shows the precision-recall curves for the two methods. To further evaluate performance

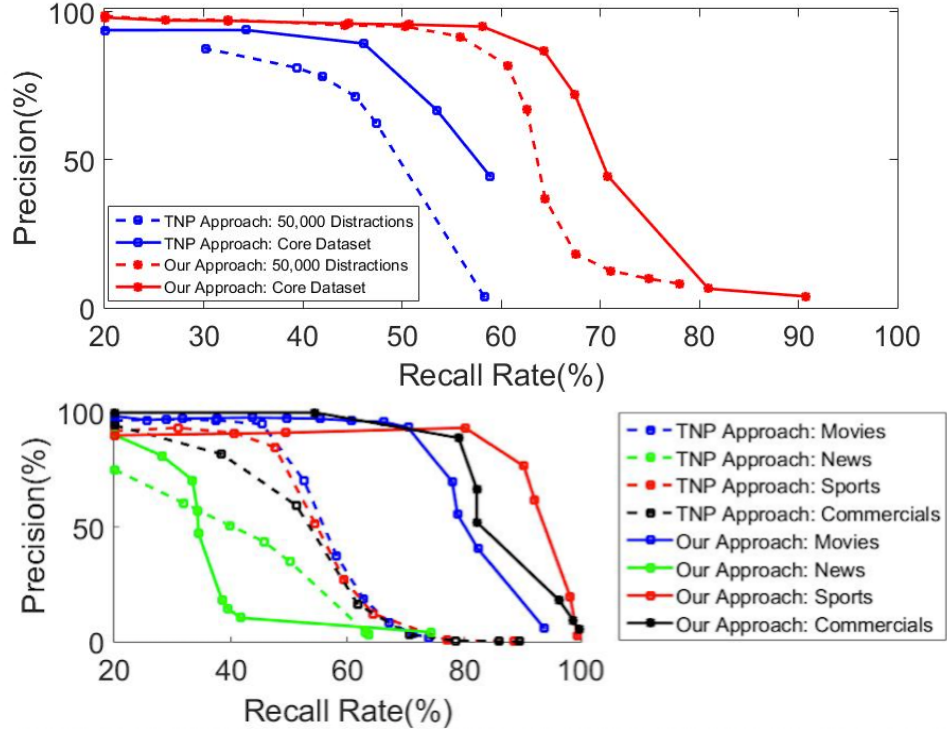


Figure 7.5: Left: the Precision-Recall curve of my approach and TNP on the VCDB +50,000 dataset., Right: The Precision-Recall curve for TNP and my approach for different genres of videos.

with respect to database size, I added 50,000 YouTube videos (total length 4300 hrs) to the VCDB dataset (precision-recall curves in Figure 7.5). This reduced my method's mean average precision to 63.76% whereas TNP's (Tan et al., 2009) mean average precision had a 15% degradation to 44.56%. Notice that my approach not only yields a 19.2% better mean average precision, but also improved recall by 18.2% over TNP (Tan et al., 2009), at the same precision of 90.0%.

Influence of Video Type: Naturally, my accuracy depends on the characteristics of the pirated content; the more illumination changes in the original material, the better. As noted earlier, Xu *et al.* (Xu et al., 2014a) observed that these illumination changes differ among categorizes of videos (e.g., movies, news and politics, sports, and commercials). To further evaluate the performance of my method, I grouped the videos in the VCDB dataset by category. The results based on this categorization are shown in Figure 7.5. My analysis shows that news clips tend to perform the worst with my approach, which is not surprising given that the news clips in VCDB are of anchormen seated at their desks covering news stories or political commentary of the day, leading to few average

intensity changes. My overall performance decreased due to the fact that 26% of the VCDB dataset are such clips. For comparison, I also evaluated the accuracy of TNP (Tan et al., 2009) considering the different grouping of videos. The results (dashed lines in Figure 7.5) show that the approach of Tan *et al.* (Tan et al., 2009) has slightly better performance on only a handful of the news videos.

Detailed Findings: I further investigated specific samples from the VCDB dataset where I succeed but the current state of the art such as TNP performs poorly. Through my empirical evaluations, I found that the temporal network approach of Tan *et al.* (Tan et al., 2009) mostly fails on detecting copies that are modified by spatial flipping, brightness adjustments, or insertion of captions — which are all important classes of spatial transformations present in real world piracy and to which my approach is robust. The sensitivity to spatial flipping could be dealt with in the techniques of Tan *et al.* (Tan et al., 2009) by flipping the uploaded video before the matching process is performed, but the illumination changes and insertion of captions would still pose significant challenges to the frame-retrieval-based framework. Specifically, the currently widely used spatial features such as SIFT and SURF provide weak performance against illumination changes (Juan and Gwun, 2009), which are 20% of the VCDB videos. I also observed a severe limitation with respect to their ability to distinguish between different types of inserted captions (8.9% of the VCDB videos). As a result, the pirated video sub-sequences with illumination changes cannot be detected, and captions copied into the pirated video lead to false detections because they are incorrectly matched to random text in the database.

Component Analysis: My method builds on the gradient feature of Xu *et al.* (Xu et al., 2014a) and provides two major improvements. First, I take temporal transformations (e.g. scaling, cropping, editing, and reversing) into consideration; second, I use a hierarchical feature leveraging a 3×3 tiling in each frame and use a classifier for copy detection based on this hierarchical feature. To better understand the contribution of each of the components of my approach with respect to Xu *et al.* (Xu et al., 2014a) and evaluate their effect on the mAP, I test it using my approach without tiling, *i.e.* only taking temporal transformations into account. I also compare it to my full approach to measure the impact of the tiling. The results are illustrated in Figure 7.6. It can be seen that both

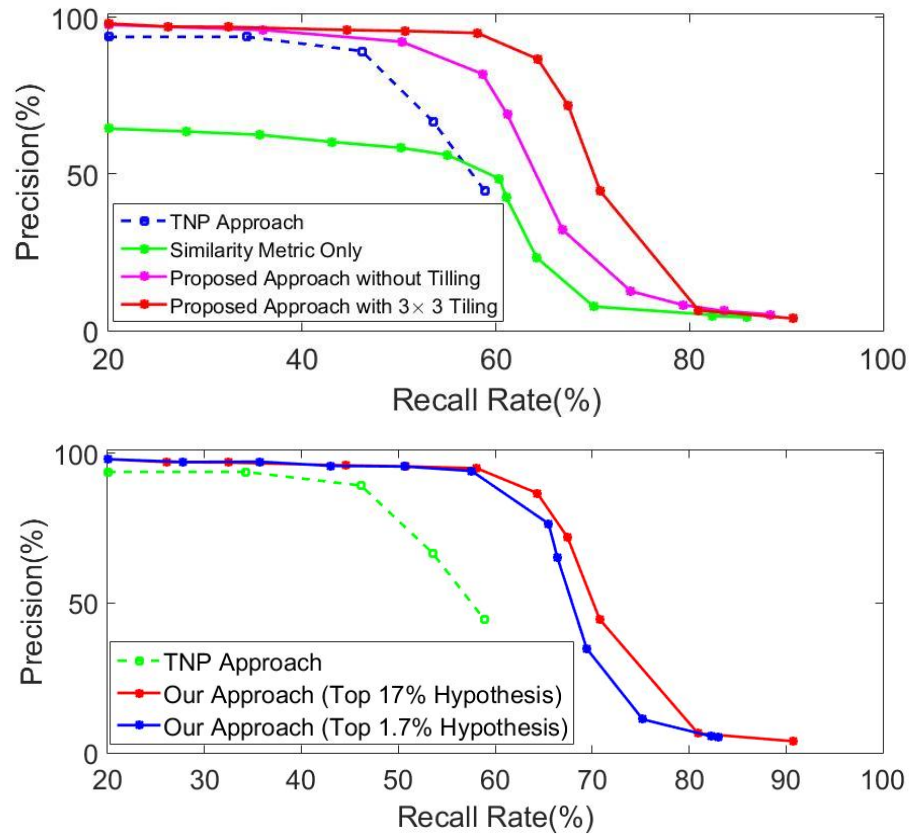


Figure 7.6: Left: Precision-Recall curve for my approach with (red)/without (purple) 3×3 tiling, Xu *et al.* (Xu et al., 2014a) (green) and TNP (blue). Right: Precision-Recall curve of the TNP approach vs my approach with different accuracies.

Temporal Scaling	50%	80%	90%	100%	110%	120%	150%
Recall Rate (My Approach)	56.0%	77.1%	84.7%	97.8%	83.3%	78.7%	58.7%
Recall Rate ((Xu et al., 2014a)’s Approach)	2.2%	5.1%	5.7%	100.0%	4.4%	2.2%	1.6%

Table 7.1: Recall rate for different temporal scaling

components of my approach significantly improve the accuracy of the copy detection. Xu *et al.*’s (Xu et al., 2014a) original work has a mAP of 40.66%, while my proposed approach without tiling reaches 63.58% (already outperforming TNP). Moreover, with the 3×3 tiling, the mean average precision of my approach further improves to 69.72%, which is significantly higher than the 60.92% of TNP approach (Tan et al., 2009). Further investigating the tiling effect, I empirically find that tiling enables us to detect copies for videos having little global brightness change, *e.g.*, sports videos from static cameras where the players cross the field. In these situations, there are no peaks for the video. However, a player crossing from one image-tiling zone into the next will create a local peak for the two zones. This enables the tiling to detect the pirated video.

Robustness Against Temporal Scaling: To demonstrate the robustness of my approach against temporal scaling, I scale the number of frames of each video in VCDB dataset from 50% (half the frames, *i.e.* doubling the speed) to 150% (adding 50% frames, *i.e.* 25% slower speed) and retrieve the adjusted video using the original video with my approach. I set parameters such that I achieve a precision of 95% and a recall of 58% for the standard VCDB database. The results are listed in Table 7.1. It can be seen that when adjusting the temporal speed with 20%, my approach is still able to detect around 78% of the copies. Even if the temporal speed is adjusted as much as 50%, my recall rate is still higher than 56%. For comparison, the recall rate of Xu *et al.* (Xu et al., 2014a)’s work completely drops below 6% when the temporal speed is adjusted as little as 8%.

Computational Overhead: While precision and recall rates are important considerations, the computational complexity of a content-based detection system is equally important, particularly given the fact that as much as 300 hours of videos are uploaded to UGC platforms per minute (Youtube, 2015). In general, the computational cost of any copy detection algorithm depends on its required accuracy, the length of the uploaded video, and the size of the reference dataset.

The bottleneck of our approach lies in the sub-sequence matching process, where we evaluate the most likely copy sub-sequence correspondences. The more correspondences we test, and the longer the uploaded video, the more computation time is required. Specifically, our proposed approach has computational complexity $O(HL)$, where H is the number of tested frame correspondences and L is the length of the uploaded video. In practice, this is also affected by other factors such as peak-rate.

Speed versus Accuracy: The design of our approach allows us to explore the tradeoff between accuracy and speed. The parameter that has the largest influence on accuracy is α , that is, the number of hypotheses that are tested during the copy detection phase. To achieve a good compromise between recall, precision, and computational cost, we examined a range of values for α . The results are shown in Figure 7.6.

Regarding speed, we note that our current implementation consists of two components: the local feature correspondence retrieval and the sub-sequence estimation. The correspondence retrieval is implemented through the VLFeat library (Vedaldi and Fulkerson, 2010) for the K-d tree search, and the sub-sequence estimation is implemented using Matlab. To study the computational overhead of our approach, we use a six-core computer with 2.40GHz CPU and 24GB memory. Our results indicate that to obtain near optimal precision, the top 17% of the hypotheses would have to be tested, which results in an average evaluation time of 70 ms per frame with a standard deviation of 4.5 ms.

The results depicted in Figure 7.6 show that we can improve computational performance by only testing the top 1.7% of the hypotheses while reducing the recall rate by only about 3%. When α is set to 1.7%, the average detection time reduces to 8.6 ms per frame with a standard deviation of 0.8 ms. These results validate that the sub-sequence estimation component of our proposed approach is linearly related to the number of tested frame correspondences. Moreover, on the 540 hrs dataset, TNP (Tan et al., 2009) requires almost twice as much time per frame (3.06s) compared to our method (1.7s). When executing our method on a database of 4,300 hours (160x VCDB core dataset size) we observe a sublinear increase of compute time (a factor of 48).

Query Length: Given that the entire query video has to be checked for potential infringement, it is obvious that the computational cost is related to the length of the uploaded video. The performance evaluation in Figure 7.7 shows that when the query length increases, the computational cost of our techniques increases modestly from 14 seconds for a 40-second video to 64-seconds for a 20 minute one. This is a dramatic improvement over the state-of-the-art system (Tan et al., 2009), whose computational time ranges from 8 minutes to nearly 4.7 hours — making it clearly impractical for real world use. It is prudent to note that our implementation of the benchmark approach follows

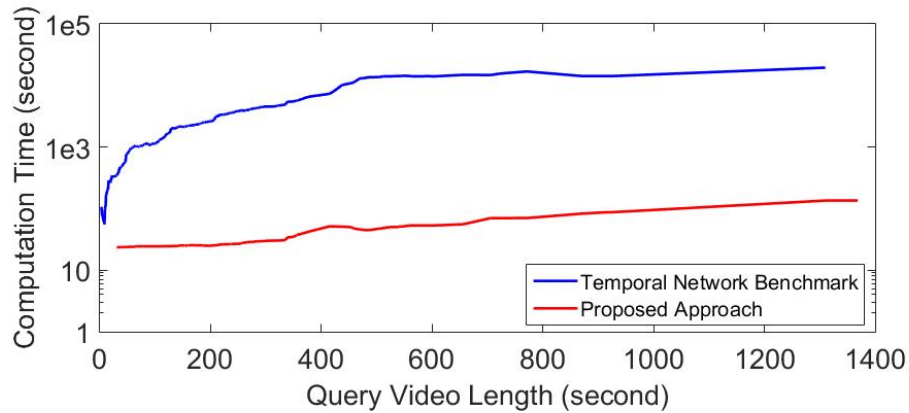


Figure 7.7: The influence of query length on computation cost

the guidelines presented by Jiang *et al.* (Jiang et al., 2014) and is consistent with the computation time of the semi-brute-force matching used by Wu *et al.* (Wu et al., 2007). The significantly higher computation time of the baseline approach can be attributed to the fact that it requires a frame retrieval stage that takes place twice for every second in the uploaded video. Tan *et al.* (Tan et al., 2005) suggest to only use the most likely candidate keyframe correspondences, which would boost the frame retrieval process by a factor of 58 at the expense of a minor loss of precision. However, our approach still outperforms theirs by an order of magnitude.

Reference library size: An increase in the size of the reference library will affect the computational performance of any CBCD approach. For our approach, a larger dataset influences efficiency in two ways: first, a larger dataset results in a longer time to retrieve the local feature correspondences, which leads to longer retrieval time; second, a larger dataset necessitates more local feature correspondences to be evaluated for copied sub-sequences, leading to higher sub-sequence estimation

times. To test the impact of increased library size, we run our proposed approach with the above described 50,000 additional videos from YouTube added to the VCDB dataset and α remaining at 1.7%. The results show that the average computational time increased from 8.6 ms per frame to 410 ms per frame. While this overhead is higher than is demanded by practice, we do believe our prototype implementation can be greatly improved (e.g., if rewritten in C). We leave that as an goal for future work.

Limitations: While my proposed approach significantly outperforms the current state of the art, there are a number of modifications that skilled adversaries could apply to degrade my copy detection rates. My technique relies on the existence of temporal gradient information. A digital pirate could temporally smooth the video’s gradient feature to hide these sudden illumination changes. Doing so would degrade the detection and retrieval performance of my method. Furthermore, my approach assumes a piecewise consistent temporal transformation. Thus, an adversary may attempt to transform the video by constantly changing the frame rate to undermine the sub-sequence detection stage and thereby impair my retrieval process. Similarly, the modifications would also thwart the current state of the art such as Tan *et al.* (Tan et al., 2009)’s TNP approach. For instance, the temporal smoothing will undermine the spatial feature extraction phase of the TNP approach, making it difficult to match a query frame and its reference. However, it remains unclear whether these are acceptable avenues for piracy in the real world. Transformations such as temporal smoothing can seriously influence the video’s visual quality, which would be at odds with the pirate’s intent of monetizing revenue generated from views of the transformed video.

7.4 Conclusion

I propose a novel method for efficiently and robustly detecting pirated video content on user-generated content platforms. My method is robust against the frequently used spatial and temporal transformations observed in the wild. The techniques I propose for enhanced subsequence matching of video-based content allows us to achieve significantly improved precision and substantially better computational performance and scalability than the current state-of-the-art approaches.

More importantly, my technique narrows the detection gap in the important area of temporal transformations applied by would-be pirates. My large-scale evaluation on real-world data shows that I can successfully detect infringing content from movies, commercials, and sports clips with 90.0% precision at a 75.1% recall rate, and can achieve this with an average time expense of merely 19 seconds, outperforming the state of the art by an order of magnitude.

CHAPTER 8: DISCUSSION

This dissertation has presented applications of video event detection and retrieval techniques under different adversary settings.

In Chapter 3, I explored the robustness and usability of MIOR CAPTCHAs, designing and implementing automated attacks based on video event detection techniques. My attack inherently leverages the temporal information in moving-image object recognition (MIOR) CAPTCHAs, and also exploits the fact that only object recognition of known objects is needed. My methods also rely on a reasonably consistent appearance or slowly varying appearance over time. That said, they can be applied to any set of known objects or narrowly defined objects under affine transformations that are known to work well with detection methods in computer vision (Viola and Jones, 2001). Looking towards the future, greater robustness would result if MIOR CAPTCHAs required automated attacks to perform classification, categorization of classes with large inner class variance, or to identify higher level semantics to understand the presented challenge. Consider, for example, the case where the user is presented with two objects (a person and a truck) at the same scale, and asked to identify which one is larger. To succeed, the automated attack would need to determine the objects (without prior knowledge of what the objects are of) and then understand the relationship. Humans can perform this task because of the inherent priors learned in daily life, but this feat remains a daunting problem in computer vision. Therefore, this combination seems to offer the right balance and underscores the ideas put forth by Naor (Naor, 1996) and von Ahn et al. (Ahn et al., 2003)—i.e., it is prudent to employ hard (and useful) underlying AI problems in CAPTCHAs since it leads to a win-win situation: either the CAPTCHA is not broken and there is a way to distinguish between humans and computers, or it is broken and a useful problem is solved.

In Chapter 4, I introduced a novel approach to bypass modern face liveness detection systems. I leveraged a handful of pictures of the target user taken from social media to create a realistic, textured, 3D facial model that undermines the security of widely used face authentication solutions. My work outlines several important lessons for both the present state and the future state of security, particularly as it relates to face authentication systems. First, my exploitation of social media photos to perform facial reconstruction underscores the notion that online privacy of one's appearance is tantamount to online privacy of other personal information, such as age and location. The ability of an adversary to recover an individual's facial characteristics through online photos is an immediate and very serious threat, albeit one that clearly cannot be completely neutralized in the age of social media. Therefore, it is prudent that face recognition tools become increasingly robust against such threats in order to remain a viable security option in the future. At a minimum, it is imperative that face authentication systems be able to reject synthetic faces with low-resolution textures, as I show in my evaluations. Of more concern, however, is the increasing threat of virtual reality, as well as computer vision, as an adversarial tool. It appears to us that the designers of face authentication systems have assumed a rather weak adversarial model wherein attackers may have limited technical skills and be limited to inexpensive materials. This practice is risky, at best. Unfortunately, VR itself is quickly becoming commonplace, cheap, and easy-to-use. Moreover, VR visualizations are increasingly *convincing*, making it easier and easier to create realistic 3D environments that can be used to fool visual security systems. As such, it is my belief that authentication mechanisms of the future must aggressively anticipate and adapt to the rapid developments in the virtual and online realms. Specifically, face authentication should rely on visual cues that cannot be easily simulated. For instance, the temporal information of stylish facial expressions (*e.g.*, breathing and pulsing patterns) can be viable alternative features. Firstly, these temporal information cannot be easily modeled and simulated by the adversary from still images of the user. Moreover, since it is already known that gait reveals the identity of a person (Stevenage et al., 1999), it is possible that these stylish expressions are also distinguishable features of the user's identity. I leave these to my future work.

In Chapter 5, I provided an approach which broadens the scope of compromising emanation attacks on mobile devices. My approach overcomes the limitation of low image resolution encountered by previous work by extending computer vision methods to operate on small, high-noise images of the mobile device, using fingertip motion for key-press detection. Specifically, I show that it is possible to perform such attacks in a number of challenging scenarios: on devices with any type of virtual or physical keyboards, without direct line-of-sight, and at distances farther away from the victim than previously thought possible. My attack can even directly use reflections on the eye-ball, which was not possible before due to the noise and physical boundaries of the optics in prior work. At the current stage, my approach does not consider the switching between alphabetical and numeric keyboard. It also does not apply to swipe-typing keyboard. I leave these possible extensions to my future work.

In Chapter 6, I introduced an attack that exploits the emanations of changes of light (e.g., as seen through the windows and recorded several meters away) to reveal the programs we watch. My attack shows reliable identification of the content being watched in a wide range of evaluated scenarios. The robustness of the attack is due to a novel application of unique feature sets, a well suited similarity metric, and the development of efficient indexing structures for performing rapid matches in near real-time. My empirical results show that I can successfully confirm hypotheses while capturing short recordings (typically less than 4 minutes long) of the changes in brightness from the victim's display. As a possible extension in future work, my proposed method could be easily applied to detecting co-occurrence in a dataset. Specifically, my work shows that the light emitted by viewing devices or any other modulated or random pattern light source (e.g., flashes of cameras) induces a distinctive flicker pattern that can be used to find linkages among videos. Therefore, I could potentially leverage these flicker patterns to identify the co-occurrence of two videos even if they have no visually overlap. In my preliminary experiments, I have already demonstrated that this is viable in controlled lab settings. However, in practice, my ability of discovering co-occurrence would be affected by camera movement and surround light that is only captured by one camera, which add strong noise peaks in the extracted temporal derivative feature. Therefore, my original

approach may need to be extended for robustness against these factors in order to be effective for co-occurrence detection in the wild. I leave these discussions to future work.

Finally, in Chapter 7, I proposed a novel method for efficiently and robustly detecting pirated video content on user-generated content platforms. My method is robust against the frequently used spatial and temporal transformations observed in the wild. The techniques I propose for enhanced subsequence matching of video-based content allows us to achieve significantly improved precision and substantially better computational performance and scalability than the current state-of-the-art approaches. More importantly, my technique narrows the detection gap in the important area of temporal transformations applied by would-be pirates. My large-scale evaluation on real-world data shows that I can successfully detect infringing content from movies, commercials, and sports clips with 90.0% precision at a 75.1% recall rate, and can achieve this with an average time expense of merely 19 seconds, outperforming the state of the art by an order of magnitude. It is also worth noting that my technique relies on the existence of temporal gradient information. A digital pirate could temporally smooth the video's gradient feature to hide these sudden illumination changes. Doing so would degrade the detection and retrieval performance of my method. Similarly, the modifications would also thwart the current state of the art such as Tan *et al.* (Tan et al., 2009)'s TNP approach. However, it remains unclear whether these are acceptable avenues for piracy in the real world. Transformations such as temporal smoothing can seriously influence the video's visual quality, which would be at odds with the pirate's intent of monetizing revenue generated from views of the transformed video. We leave these analysis to future work.

To summarize, in Chapter 3-6, I mainly explores the vulnerabilities of existing security systems (Chapter 3, Chapter 4) and possible side-channel attacks (Chapter 5, Chapter 6). By exploring these security flaws, researchers have better understanding of the adversary's constraints, and mitigate these attacks to enhance the security of existing systems. On the other hand, in Chapter 7, I think from the defender's perspective and proposed a novel method for content-based copy detection using video retrieval techniques. In my approaches above, I do not simply integrate well developed computer vision techniques. Instead, I extend the boundary of video event detection and retrieval

techniques to better fitting the adversary constraints, making contributions in the computer vision domain.

APPENDIX

A Parameters for video generation

Similar to NuCaptcha’s videos, my sequences have letters that move across a background scene with constant velocity in the horizontal direction, and move up and down harmonically (i.e., $y(t) = A * \sin(\omega t + \psi)$, y is the vertical position of the letter, t is the frame id, and A, ω, ψ are adjustable parameters). The horizontal distance between two letters is a function of their average width. If their widths are $width_1, width_2$, the distance between their centers are set to be $\alpha * \frac{width_1 + width_2}{2}$, where α is an adjustable parameter that indicates how much two letters overlap. My letters also rotate and loop around. The angle θ to which a letter rotates is also decided by a sin function $\theta = \theta_0 * \sin(\omega_\theta t + \psi_\theta)$, where $\theta_0, \omega_\theta, \psi_\theta$ are adjustable parameters. For the standard case, I set the parameters the same as in NuCaptcha’s videos. I adjust these parameters based on the type of defenses I explore (in Section 3.3.2).

B Comments from User Study

Table B1 highlights some of the free-form responses written on the questionnaire used in my study.

C Multi-Image Facial Model Estimation

In §4.2.2, I outline how to associate 2D facial landmarks with corresponding 3D points on an underlying facial model. Contour landmarks pose a substantial difficulty for this 2D-to-3D correspondence problem because the associated set of 3D points for these features is pose-dependent. Zhu et al. (2015) compensate for this phenomenon by modeling contour landmarks with parallel curved line segments and iteratively optimizing head orientation and 2D-to-3D correspondence. For a specific head orientation R_j , the corresponding landmark points on the 3D model are found using

Variant	Comments
<i>Standard</i>	<ul style="list-style-type: none"> - User friendly - It was too easy - Much easier than traditional CAPTCHAs
<i>Extended</i>	<ul style="list-style-type: none"> - My mother would not be able to solve these - Giant Pain in the Butt! Sheer mass of text was overwhelming and I got lost many times - Too long! I would prefer a shorter text - It was very time consuming, and is very prone to mistakes
<i>Overlapping</i>	<ul style="list-style-type: none"> - Letters too bunched – several loops needed to decipher - Takes longer because I had to wait for the letter to move a bit so I can see more of it - Still had a dizzying affect. Not pleasant - Some characters were only partially revealed, ‘Y’ looked like a ‘V’
<i>Semi-Transparent</i>	<ul style="list-style-type: none"> - Tree background is unreadable, any non-solid background creates too much interference - With some backgrounds I almost didn’t realize there were red letters - It was almost faded and very time consuming. I think I made more mistakes in this mechanism
<i>Emerging</i>	<ul style="list-style-type: none"> - Not that complicated - I’d feel dizzy after staring at it for more than 1 min - It was hideous! Like an early 2000s website. But it did do the job. It made my eyes feel ‘fuzzy’ after a while - It was good, better than the challenges with line through letters

Table B1: Sample participant comments for each variant

an explicit function based on rotation angle:

$$\begin{aligned}
s_{i,j} &= f_j PR_j(S_{i',j} + t_j) \\
S_{i',j} &= \bar{S}_{i'} + A_{i'}^{id} \alpha^{id} + A_{i'}^{exp} \alpha_j^{exp} \\
i' &= land(i, R_j),
\end{aligned} \tag{1}$$

where $land(i, R_j)$ is the pre-calculated mapping function that computes the position of landmarks i on the 3D model when the orientation is R_j . Ideally, the first equation in Eq. (1) should hold for all the landmark points in all the images. However, this is not the case due to the alignment error introduced by landmark extraction. Generally, contour landmarks introduce more error than corner landmarks, and this approach actually leads to inferior results when multiple input images are used.

Therefore, different from Zhu et al. (2015), I compute the 3D facial model with Maximum a Posteriori (MAP) estimation. I assume the alignment error of each 3D landmark independently follows

a Gaussian distribution. Then, the most probable parameters $\theta := (\{f_j\}, \{R_j\}, \{t_j\}, \{\alpha_j^{exp}\}, \alpha^{id})$ can be estimated by minimizing the cost function

$$\begin{aligned} \theta = \operatorname{argmin}_{\theta} \{ & \sum_{i=1}^{68} \sum_{j=1}^N \frac{1}{(\sigma_i^s)^2} \|s_{i,j} - f_j P R_j (S_{i',j} + t_j)\|^2 + \\ & \sum_{j=1}^N (\alpha_j^{exp})' \Sigma_{exp}^{-1} \alpha_j^{exp} + (\alpha^{id})' \Sigma_{id}^{-1} \alpha^{id} \}. \end{aligned} \quad (2)$$

Here, $S_{i',j}$ is computed using Eq. (1). Σ_{id} and Σ_{exp} are covariance matrices of α^{id} and α_j^{exp} , which can be obtained from the pre-existing face model. $(\sigma_i^s)^2$ is the variance of alignment error of the i -th landmark and is obtained from a separate training set consisting 20 images with hand-labeled landmarks. Eq. (2) can be computed efficiently, leading to the estimated identity weight α^{id} , with which I can compute the neutral-expression model $S_i (= \bar{S}_{i'} + A_{i'}^{id} \alpha^{id})$.

D Anisotropic Diffusion

My alignment to the reference layout (Section 5.2) is based on image edges. Hence I opt for anisotropic diffusion, which is a noise suppression algorithm that maintains edges in the image. My anisotropic diffusion is in the spirit of the method proposed by Weickert (Weickert, 1998), which was designed to preserve all detected edges in the image. In my work however, I only preserve images edges that are along the dominant directions of the boundaries of the mobile device. This not only reduces noise but also effectively suppresses spurious lines, as for example, caused by reflections on the screen. The diffusion process is described by:

$$u' = \begin{cases} \partial_t u = \operatorname{div}((1 - c(\phi)(1 - D(J_\rho(\nabla u_\sigma)))) \nabla u) \\ c(\theta) = K_\Sigma(\phi - \operatorname{Dir}_1) + K_\Sigma(\phi - \operatorname{Dir}_2) \\ D(J_\rho(\nabla u_\sigma)) = U^T \begin{pmatrix} 1/(1 + \lambda_1) & 0 \\ 0 & 1 \end{pmatrix} U \end{cases} \quad (3)$$

where u is the pixel value of the original image, u' is the new pixel value, and ϕ is the angle of the eigenvector of the biggest eigenvalue λ_1 of J_ρ (the Jacobian matrix of the image at pixel u). The matrix U is the unitary matrix consisting of the eigenvectors of J_ρ . K_Σ is a Gaussian kernel with parameter Σ , and Dir_1, Dir_2 are the pre-calculated dominant directions. The resulting filter output has the following properties:

- 1) If the edge is strongly deviating from the main direction (i.e. has a large angle to the dominant direction), it will be blurred since Equation (3) converges to $\partial_t u = \text{div}(\nabla u)$.
- 2) Edges along the dominant directions will be enhanced along the edge since $D(J_\rho(\nabla u_\sigma))$ becomes very small. This means the edge will not be blurred and will maintain its high contrast.
- 3) In addition, the areas alongside the edges in the dominant directions will be smoothed in the direction parallel to the edge to suppress the noise since in this direction I also have $\partial_t u = \text{div}(\nabla u)$.

In summary this will ensure an effective noise suppression while maintaining edges in the dominant directions, which are then used to align the device image to the reference layout.

E Vanishing Points

Vanishing points are points in the 2D projected image where lines that are parallel converge. There is a large body of work on estimating vanishing points. Most of the methods are based on pre-marked points and mainly aim to solve problems when there is significant skew in the directions of the lines, such as (Magee and Aggarwal, 1984). However, these methods fail in my application because the vanishing points are so far away that the data matrix becomes singular. Alternative methods are based on transformations such as the Hough transform (Lutton et al., 1994) and Sphere representation (Collins and Weiss, 1990).

Due to its ability to detect multiple vanishing points at once I chose to use a Hough transform to calculate the vanishing points. The Hough transform is a voting scheme in the parameter space where every pixel along a line segment votes for all compatible parameterizations of its line. The detected lines are then chosen as the local maxima in the voting space. I use the angular representation $x\cos\theta + y\sin\theta = \rho$ of line segments for the voting, which avoids the singularity

of vertical lines in the slope and bias parameterization of lines. The Hough transformation then provides the θ_i, ρ_i of all lines l_i in the image. These lines are then used to compute the vanishing point $v = (x, y)$ through solving:

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} \rho \frac{d\theta}{d\rho} \cos \theta - \sin \theta \\ \rho \frac{d\rho}{d\theta} \sin \theta + \cos \theta \end{pmatrix} * \frac{1}{d\theta/d\rho} \quad (4)$$

for each of the lines. To combine the information from all lines along each of the major directions I use regression to compute $d\theta/d\rho$ and θ . This is then used in solving for the vanishing points (x, y) with Equation (4), which results in the horizontal vanishing point v_x and the vertical vanishing point v_y . Assuming v_x, v_y are normalized to satisfy $\|v_x\| = \|v_y\| = 1$ the camera rotation matrix R is given by

$$R = \begin{pmatrix} v_x \\ v_y \\ v_x \times v_y \end{pmatrix}$$

Applying the inverse rotation ensures the device screen plane and the coordinate systems x - y -plane are parallel, effectively removing any rotation-related distortion from the image.

F Translation Alignment

The transformation of a given image to the template can be viewed as rotation of the camera followed by translation and zoom. My method uses the Hough transformation to extract the camera's rotation, translation, and scale as explained in Section 5.2. Here, I introduce how to compute the resulting transformation given estimated parameters.

The vanishing points reveal the rotation information. For instance, all the horizontal lines on the device correspond to a vanishing point (x, y) on the image plane, which is calculated as described above. The direction of these lines can be calculated as the direction from the center of the camera to the pixel (x, y) . For a pinhole camera, this 3D direction *Direction* can be represented as: $(x - c_x, y - c_y, f)^T$ with (c_x, c_y) being the coordinate of the center of the image and f representing

the focal length of the camera normalized by the physical size of the pixels. These are intrinsic parameters modeling the imaging parameters of the camera sensor and lens.

With the vanishing points, I acquire the two dominant directions of the edges of the device as $Direction_h = (x_h, y_h, z_h)$ and $Direction_v = (x_v, y_v, z_v)$. The edges of the device can be transformed to become horizontal and vertical with following equation:

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \frac{\begin{pmatrix} \frac{x-c_x}{f}x_h + \frac{y-c_y}{f}y_h + z_h \\ \frac{x-c_x}{f}x_v + \frac{y-c_y}{f}y_v + z_v \end{pmatrix} * f}{\det \begin{pmatrix} x_h & y_h & z_h \\ x_v & y_v & z_v \\ \frac{x-c_x}{f} & \frac{y-c_y}{f} & 1 \end{pmatrix}} + \begin{pmatrix} c_x \\ c_y \end{pmatrix}$$

Here, point (x, y) in the image is transformed to (x', y') . This equation virtually rotates the camera in the 3D space so that the coordinate systems x - y -plane is parallel with the device's boundary. As discussed in Section 5.2, the translation parameters are calculated using a Hough transform using the weighted line voting. Considering the parameters acquired by the line matching (translations t_x, t_y and scaling s_x, s_y in x -direction and y -direction respectively), the transformation performing translation and scale alignment is given as: $(x'', y'')^T = (s_x x' + t_x, s_y y' + t_y)^T$. This maps the previous transformation result (x', y') to the point (x'', y'') . In this way, I successfully transform the device screen's image into the template's coordinate system.

BIBLIOGRAPHY

- Aggarwal, J. K., Cai, Q., Liao, W., and Sabata, B. (1994). Articulated and elastic non-rigid motion: A review. In *Motion of Non-Rigid and Articulated Objects, 1994., Proceedings of the 1994 IEEE Workshop on*, pages 2–14. IEEE.
- Ahn, L. V., Blum, M., Hopper, N. J., and Langford, J. (2003). CAPTCHA: using hard AI problems for security. In *Eurocrypt*, pages 294–311.
- Arxan Technologies (2015). <https://www.arxan.com/wp-content/uploads/2015/06/state-of-application-security-report-vol-4-2015.pdf>.
- Aslandogan, Y. A. and Yu, C. T. (1999). Techniques and systems for image and video retrieval. *Knowledge and Data Engineering, IEEE Transactions on*, 11(1):56–63.
- Asonov, D. and Agrawal, R. (2004). Keyboard acoustic emanations. In *Proceedings of the IEEE Symposium on Security and Privacy*.
- Backes, M., Chen, T., Dürmuth, M., Lensch, H., and Welk, M. (2009). Tempest in a teapot: Compromising reflections revisited. In *Proceedings of the IEEE Symposium on Security and Privacy*.
- Backes, M., Dürmuth, M., and Unruh, D. (2008). Compromising reflections-or-how to read LCD monitors around the corner. In *Proceedings of the IEEE Symposium on Security and Privacy*.
- Baker, S. and Matthews, I. (2004). Lucas-kanade 20 years on. *International Journal of Computer Vision*, 56(3):221–255.
- Balakrishnan, G., Durand, F., and Guttag, J. (2013). Detecting pulse from head motions in video. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3430–3437.
- Balzarotti, D., Cova, M., and Vigna, G. (2008a). Clearshot: Eavesdropping on keyboard input from video. In *Security and Privacy, 2008. SP 2008. IEEE Symposium on*, pages 170–183. IEEE.
- Balzarotti, D., Cova, M., and Vigna, G. (2008b). ClearShot: Eavesdropping on keyboard input from video. In *Proceedings of the IEEE Symposium on Security and Privacy*.
- Bao, W., Li, H., Li, N., and Jiang, W. (2009). A liveness detection method for face recognition based on optical flow field. In *Image Analysis and Signal Processing, International Conference on*, pages 233–236.
- Bartholomew, T. B. (2014). Death of fair use in cyberspace: Youtube and the problem with content id, the. *Duke L. & Tech. Rev.*, 13:66.
- Basso, A. and Bergadano, F. (2010). Anti-bot strategies based on human interactive proofs. In Stavroulakis, P. and Stamp, M., editors, *Handbook of Information and Communication Security*, pages 273–291. Springer.

- Baumberger, C., Reyes, M., Constantinescu, M., Olariu, R., De Aguiar, E., and Oliveira Santos, T. (2014). 3d face reconstruction from video using 3d morphable model and silhouette. In *Graphics, Patterns and Images (SIBGRAPI), Conference on*, pages 1–8.
- Belhumeur, P. N., Jacobs, D. W., Kriegman, D. J., and Kumar, N. (2013). Localizing parts of faces using a consensus of exemplars. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 35(12):2930–2940.
- Bentley, J. L. (1975). Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517.
- Bhardwaj, D. and Mahajan, S. (2015). Review paper on automated number plate recognition techniques.
- Blanz, V. and Vetter, T. (1999). A morphable model for the synthesis of 3d faces. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 187–194. ACM Press/Addison-Wesley Publishing Co.
- Blanz, V. and Vetter, T. (2003). Face recognition based on fitting a 3d morphable model. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 25(9):1063–1074.
- Buades, A., Coll, B., and Morel, J.-M. (2005). A review of image denoising algorithms, with a new one. *Multiscale Modeling & Simulation*, 4(2):490–530.
- Bursztein, E. (2012). How we broke the NuCaptcha video scheme and what we proposed to fix it. See <http://elie.im/blog/security/how-we-broke-the-nucaptcha\discretionary{-}{ }{ }{ }video-scheme-and-what-we-propose-to-fix-it/>.
- Bursztein, E., Beauxis, R., Paskov, H., Perito, D., Fabry, C., and Mitchell, J. C. (2011a). The failure of noise-based non-continuous audio CAPTCHAs. In *IEEE Symposium on Security and Privacy*, pages 19–31.
- Bursztein, E. and Bethard, S. (2009). DeCAPTCHA: breaking 75% of eBay audio CAPTCHAs. In *Proceedings of the 3rd USENIX Workshop on Offensive Technologies*.
- Bursztein, E., Bethard, S., Fabry, C., Mitchell, J. C., and Jurafsky, D. (2010). How good are humans at solving CAPTCHAs? a large scale evaluation. In *IEEE Symposium on Security and Privacy*, pages 399–413.
- Bursztein, E., Martin, M., and Mitchell, J. (2011b). Text-based CAPTCHA strengths and weaknesses. In *ACM Conference on Computer and Communications Security*, pages 125–138.
- Cai, L. and Chen, H. (2011). Touchlogger: inferring keystrokes on touch screen from smartphone motion. In *USENIX Workshop on Hot Topics in Security (HotSec)*.
- Cai, L. and Chen, H. (2012). On the practicality of motion based keystroke inference attack. *Trust and Trustworthy Computing*, pages 273–290.

- Cao, C., Weng, Y., Zhou, S., Tong, Y., and Zhou, K. (2014). Facewarehouse: A 3d facial expression database for visual computing. *Visualization and Computer Graphics, IEEE Transactions on*, 20(3):413–425.
- Cedras, C. and Shah, M. (1994). A survey of motion analysis from moving light displays. In *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR'94., 1994 IEEE Computer Society Conference on*, pages 214–221. IEEE.
- Cedras, C. and Shah, M. (1995). Motion-based recognition a survey. *Image and Vision Computing*, 13(2):129–155.
- Chaudhary, A., Raheja, J., and Raheja, S. (2012). A vision based geometrical method to find fingers positions in real time hand gesture recognition. *Journal of Software*, 7(4):861–869.
- Chellapilla, K., Larson, K., Simard, P. Y., and Czerwinski, M. (2005a). Building segmentation based human-friendly human interaction proofs (hips). In *Human Interactive Proofs, Second International Workshop*, pages 1–26.
- Chellapilla, K., Larson, K., Simard, P. Y., and Czerwinski, M. (2005b). Designing human friendly human interaction proofs (HIPs). In *ACM Conference on Human Factors in Computing Systems*, pages 711–720.
- Chellappa, R., Wilson, C. L., and Sirohey, S. (1995). Human and machine recognition of faces: A survey. *Proceedings of the IEEE*, 83(5):705–741.
- Chen, L. and Stentiford, F. (2008). Video sequence matching based on temporal ordinal measurement. *Pattern Recognition Letters*, 29(13):1824–1831.
- Chen, T. and Kan, M.-Y. (2011). Creating a live, public short message service corpus: The NUS SMS corpus. *Language Resources and Evaluation*.
- Chenot, J.-H. and Daigneault, G. (2014). A large-scale audio and video fingerprints-generated database of tv repeated contents. In *Content-Based Multimedia Indexing (CBMI), 2014 12th International Workshop on*, pages 1–6. IEEE.
- Chikkerur, S., Sundaram, V., Reisslein, M., and Karam, L. J. (2011). Objective video quality assessment methods: A classification, review, and performance comparison. *Broadcasting, IEEE Transactions on*, 57(2):165–182.
- Chiu, C.-Y., Chen, C.-S., and Chien, L.-F. (2008). A framework for handling spatiotemporal variations in video copy detection. *Circuits and Systems for Video Technology, IEEE Transactions on*, 18(3):412–417.
- Chiu, C.-Y., Yang, C.-C., and Chen, C.-S. (2007). Efficient and effective video copy detection based on spatiotemporal analysis. In *Multimedia, 2007. ISM 2007. Ninth IEEE International Symposium on*, pages 202–209. IEEE.
- Chu, B., Romdhani, S., and Chen, L. (2014). 3d-aided face recognition robust to expression and pose variations. In *Computer Vision and Pattern Recognition (CVPR), Conference on*, pages 1907–1914.

- Collins, R. and Weiss, R. (1990). Vanishing point calculation as a statistical inference on the unit sphere. In *Computer Vision, 1990. Proceedings, Third International Conference on*, pages 400–403. IEEE.
- Cui, J., Zhang, W., Peng, Y., Liang, Y., Xiao, B., Mei, J., Zhang, D., and Wang, X. (2010a). A 3-layer Dynamic CAPTCHA Implementation. In *Workshop on Education Technology and Computer Science*, volume 1, pages 23–26.
- Cui, J.-S., Mei, J.-T., Wang, X., Zhang, D., and Zhang, W.-Z. (2009). A CAPTCHA Implementation Based on 3D Animation. In *International Conference on Multimedia Information Networking and Security*, volume 2, pages 179–182.
- Cui, J.-S., Mei, J.-T., Zhang, W.-Z., Wang, X., and Zhang, D. (2010b). A CAPTCHA Implementation Based on Moving Objects Recognition Problem. In *International Conference on E-Business and E-Government*, pages 1277–1280.
- Danaher, B. and Smith, M. D. (2014). Gone in 60 seconds: the impact of the megapupload shutdown on movie sales. *International Journal of Industrial Organization*, 33:1–8.
- Derpanis, K. G., Sizintsev, M., Cannons, K., and Wildes, R. P. (2010). Efficient action spotting based on a spacetime oriented structure representation. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 1990–1997. IEEE.
- DiCarlo, J. J. and Cox, D. D. (2007). Untangling invariant object recognition. *Trends in Cognitive Sciences*, 11:333–341.
- Dollar, P., Wojek, C., Schiele, B., and Perona, P. (2012). Pedestrian detection: An evaluation of the state of the art. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 34(4):743–761.
- Driver, J. and Baylis, G. (1996). Edge-assignment and figure-ground segmentation in short-term visual matching. *Cognitive Psychology*, 31:248–306.
- Duc, N. and Minh, B. (2009). Your face is not your password. In *Black Hat Conference*, volume 1.
- Duda, R. O. and Hart, P. E. (1972). Use of the Hough transformation to detect lines and curves in pictures. *Communications of the ACM*, 15(1):11–15.
- Eddy, S. R. (1996). Hidden markov models. *Current opinion in structural biology*, 6(3):361–365.
- Efros, A., Berg, A. C., Mori, G., Malik, J., et al. (2003). Recognizing action at a distance. In *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, pages 726–733. IEEE.
- Egele, M., Bilge, L., Kirda, E., and Kruegel, C. (2010). Captcha smuggling: hijacking web browsing sessions to create captcha farms. In *ACM Symposium on Applied Computing*, pages 1865–1870.
- Elibol, F., Sarac, U., and Erer, I. (2012). Realistic eavesdropping attacks on computer displays with low-cost and mobile receiver system. In *Proceedings of the 20th European Signal Processing Conference*.

- Elson, J., Douceur, J. R., Howell, J., and Saul, J. (2007). Asirra: a CAPTCHA that exploits interest-aligned manual image categorization. In *Proceedings of the ACM Conference on Computer and Communications Security*, pages 366–374.
- Enev, M., Gupta, S., Kohno, T., and Patel, S. N. (2011). Televisions, video privacy, and powerline electromagnetic interference. In *Proceedings of the 18th ACM conference on Computer and communications security*, pages 537–550. ACM.
- Erdogmus, N. and Marcel, S. (2014). Spoofing face recognition with 3d masks. *Information Forensics and Security, IEEE Transactions on*, 9(7):1084–1097.
- Esmaeili, M. M., Fatourehchi, M., and Ward, R. K. (2011). A robust and fast video copy detection system using content-based fingerprinting. *Information Forensics and Security, IEEE Transactions on*, 6(1):213–226.
- Faloutsos, C., Ranganathan, M., and Manolopoulos, Y. (1994). Fast subsequence matching in time-series databases. *ACM International Conference on Management of Data (SIGMOD)*, 23(2).
- Fidaleo, D. and Medioni, G. (2007). Model-assisted 3d face reconstruction from video. In *Analysis and modeling of faces and gestures*, pages 124–138. Springer.
- Fiona, A. H. Y. (2006). *Keyboard acoustic triangulation attack*. PhD thesis, Citeseer.
- Fischler, M. and Bolles, R. (1981a). Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Comm. of the ACM*, 24(6):381–395.
- Fischler, M. and Bolles, R. (1981b). Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395.
- Francis, W. N. and Kucera, H. (1979). Brown corpus manual. Technical report, Dept. of Linguistics, Brown University.
- Friedman, N. and Russell, S. (1776). Image segmentation in video sequences: A probabilistic approach. *University of California, Berkeley*, 94720.
- Gartner (2014). Gartner backs biometrics for enterprise mobile authentication. *Biometric Technology Today*.
- Gengembre, N., Berrani, S.-A., and Lechat, P. (2008). Adaptive similarity search in large databases-application to image/video copy detection. In *Content-Based Multimedia Indexing, 2008. CBMI 2008. International Workshop on*, pages 496–503. IEEE.
- Geronimo, D., Lopez, A. M., Sappa, A. D., and Graf, T. (2009). Survey of pedestrian detection for advanced driver assistance systems. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (7):1239–1258.
- Godin, C. and Lockwood, P. (1989). Dtw schemes for continuous speech recognition: a unified view. *Computer Speech & Language*, 3(2):169–198.

- Golder, S. (2008). Measuring social networks with digital photograph collections. In *Proceedings of the nineteenth ACM conference on Hypertext and hypermedia*, pages 43–48.
- Golle, P. (2008). Machine learning attacks against the Asirra CAPTCHA. In *ACM Conference on Computer and Communications Security*, pages 535–542.
- Gomery, D. (1993). As the dial turns. *The Wilson Quarterly*, pages 41–46.
- Grabner, H., Grabner, M., and Bischof, H. (2006). Real-time tracking via on-line boosting. In *Proceedings of the British Machine Vision Conference*, volume 1, pages 47–56.
- Greveler, U., Justus, B., and Loehr, D. (2012). Multimedia content identification through smart meter power usage profiles. *Computers, Privacy and Data Protection*.
- Grill-Spector, K. and Kanwisher, N. (2005). Visual recognition: as soon as you know it is there, you know what it is. *Psychological Science*, 16(2):152–160.
- Hampapur, A., Hyun, K., and Bolle, R. M. (2001). Comparison of sequence matching techniques for video copy detection. In *Electronic Imaging 2002*, pages 194–201. International Society for Optics and Photonics.
- Harris, C. and Stephens, M. (1988). A combined corner and edge detection. In *Proceedings of The Fourth Alvey Vision Conference*, volume 15, pages 147–151.
- Healey, G. E. and Kondepudy, R. (1994). Radiometric ccd camera calibration and noise estimation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 16(3):267–276.
- Hicks, M. (2011). A continued commitment to security.
- Hidalgo, J. M. G. and Alvarez, G. (2011). CAPTCHAs: An Artificial Intelligence Application to Web Security. *Advances in Computers*, 83:109–181.
- Highland, H. J. (1986). Electromagnetic radiation revisited. *Computer Security*, 5:85–93.
- Hoanca, B. and Mock, K. J. (2008). Password entry scheme resistant to eavesdropping. In *Security and Management*.
- Horaud, R., Dornaika, F., and Lamiroy, B. (1997). Object pose: The link between weak perspective, paraperspective, and full perspective. *International Journal of Computer Vision*, 22(2):173–189.
- Horn, B. K. and Schunck, B. G. (1981). Determining optical flow. In *1981 Technical symposium east*, pages 319–331. International Society for Optics and Photonics.
- Hua, X.-S., Chen, X., and Zhang, H.-J. (2004). Robust video signature based on ordinal measure. In *Image Processing, 2004. ICIP'04. 2004 International Conference on*, volume 1, pages 685–688. IEEE.
- Ikizler, N. and Forsyth, D. (2007). Searching video for complex activities with finite state models. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pages 1–8. IEEE.

- İkizler, N. and Forsyth, D. A. (2008). Searching for complex human activities with no visual examples. *International Journal of Computer Vision*, 80(3):337–357.
- Ilia, P., Polakis, I., Athanasopoulos, E., Maggi, F., and Ioannidis, S. (2015). Face/off: Preventing privacy leakage from photos in social networks. In *Proceedings of the 22nd ACM Conference on Computer and Communications Security*, pages 781–792.
- Intel Security (2015). True Key by Intel Security: Security white paper 1.0.
- Iturbe, X., Altuna, A., Ruiz de Olano, A., and Martinez, I. (2008). VHDL described finger tracking system for real-time human-machine interaction. In *International Conference on Signals and Electronic Systems*.
- Jain, A., Murty, M., and Flynn, P. (1999). Data clustering: a review. *ACM computing surveys (CSUR)*, 31(3):264–323.
- Jain, A. K. and Kumar, A. (2010). Biometrics of next generation: An overview. *Second Generation Biometrics*, 12(1):2–3.
- Jazayeri, A., Cai, H., Zheng, J. Y., and Tuceryan, M. (2011). Vehicle detection and tracking in car video based on motion model. *Intelligent Transportation Systems, IEEE Transactions on*, 12(2):583–595.
- Jee, H.-K., Jung, S.-U., and Yoo, J.-H. (2006). Liveness detection for embedded face recognition system. *International Journal of Biological and Medical Sciences*, 1(4):235–238.
- Jeni, L. A., Cohn, J. F., and Kanade, T. (2015). Dense 3d face alignment from 2d videos in real-time. In *Automatic Face and Gesture Recognition (FG), 2015 11th IEEE International Conference and Workshops on*, volume 1, pages 1–8. IEEE.
- Jesorsky, O., Kirchberg, K. J., and Frischholz, R. W. (2001). Robust face detection using the hausdorff distance. In *Audio-and video-based biometric person authentication*, pages 90–95. Springer.
- Jiang, H., Drew, M. S., and Li, Z.-N. (2006). Successive convex matching for action detection. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 2, pages 1646–1653. IEEE.
- Jiang, H., Drew, M. S., and Li, Z.-N. (2010). Action detection in cluttered video with successive convex matching. *Circuits and Systems for Video Technology, IEEE Transactions on*, 20(1):50–64.
- Jiang, Y.-G., Jiang, Y., and Wang, J. (2014). Vcdb: A large-scale database for partial copy detection in videos. In *Computer Vision–ECCV 2014*, pages 357–371. Springer.
- Jiang, Y.-G., Ngo, C.-W., and Yang, J. (2007). Towards optimal bag-of-features for object categorization and semantic video retrieval. In *Proceedings of the 6th ACM international conference on Image and video retrieval*, pages 494–501. ACM.

- Jin, L., Yang, D., Zhen, L., and Huang, J. (2007). A novel vision-based finger-writing character recognition system. *Journal of Circuits, Systems, and Computers*, 16(03):421–436.
- Jolliffe, I. (2002). *Principal component analysis*. Wiley Online Library.
- Joly, A., Buisson, O., and Frelicot, C. (2007). Content-based copy retrieval using distortion-based probabilistic similarity search. *Multimedia, IEEE Transactions on*, 9(2):293–306.
- Jones, S. and Shao, L. (2013). Content-based retrieval of human actions from realistic video databases. *Information Sciences*, 236:56–65.
- Juan, L. and Gwun, O. (2009). A comparison of sift, pca-sift and surf. *International Journal of Image Processing (IJIP)*, 3(4):143–152.
- Karu, K. and Jain, A. K. (1996). Fingerprint classification. *Pattern recognition*, 29(3):389–404.
- Ke, Y., Sukthankar, R., and Hebert, M. (2007). Event detection in crowded videos. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–8. IEEE.
- Ke, Y., Sukthankar, R., and Hebert, M. (2010). Volumetric features for video event detection. *International journal of computer vision*, 88(3):339–362.
- Ke, Y., Sukthankar, R., and Huston, L. (2004). Efficient near-duplicate detection and sub-image retrieval. In *ACM Multimedia*, volume 4, page 5.
- Kemelmacher-Shlizerman, I. (2013). Internet based morphable model. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3256–3263.
- Kemelmacher-Shlizerman, I. and Basri, R. (2011). 3D face reconstruction from a single image using a single reference face shape. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 33(2):394–405.
- Kerdvibulvech, C. and Saito, H. (2007). Vision-based detection of guitar players’ fingertips without markers. In *Computer Graphics, Imaging and Visualisation*.
- Kim, G., Eum, S., Suhr, J. K., Kim, D. I., Park, K. R., and Kim, J. (2012a). Face liveness detection based on texture and frequency analyses. In *Biometrics (ICB), 5th IAPR International Conference on*, pages 67–72.
- Kim, H.-N., El Saddik, A., and Jung, J.-G. (2012b). Leveraging personal photos to inferring friendships in social network services. *Expert Systems with Applications*, 39(8):6955–6966.
- Kim, S., Yu, S., Kim, K., Ban, Y., and Lee, S. (2013). Face liveness detection using variable focusing. In *Biometrics (ICB), 2013 International Conference on*, pages 1–6.
- Kluever, K. A. and Zanibbi, R. (2009). Balancing usability and security in a video CAPTCHA. In *Symposium on Usable Privacy and Security*, pages 1–14.
- Ko, T. (2008). A survey on behavior analysis in video surveillance for homeland security applications. In *Applied Imagery Pattern Recognition Workshop, 2008. AIPR’08. 37th IEEE*, pages 1–8. IEEE.

- Kolev, K., Tanskanen, P., Speciale, P., and Pollefeys, M. (2014). Turning mobile phones into 3d scanners. In *Computer Vision and Pattern Recognition (CVPR), IEEE Conference on*, pages 3946–3953.
- Kollreider, K., Fronthaler, H., and Bigun, J. (2005). Evaluating liveness by face images and the structure tensor. In *Automatic Identification Advanced Technologies, Fourth IEEE Workshop on*, pages 75–80. IEEE.
- Kollreider, K., Fronthaler, H., and Bigun, J. (2008). Verifying liveness by multiple experts in face biometrics. In *Computer Vision and Pattern Recognition Workshops, IEEE Computer Society Conference on*, pages 1–6.
- Kollreider, K., Fronthaler, H., Faraj, M. I., and Bigun, J. (2007). Real-time face detection and motion analysis with application in liveness assessment. *Information Forensics and Security, IEEE Transactions on*, 2(3):548–558.
- Küçüktunç, O., Baştan, M., Güdükbay, U., and Ulusoy, Ö. (2010). Video copy detection using multiple visual cues and mpeg-7 descriptors. *Journal of Visual Communication and Image Representation*, 21(8):838–849.
- Kuhn, M. and Kuhn, C. (2003). Compromising emanations: eavesdropping risks of computer displays.
- Kuhn, M. G. (2013). Compromising Emanations of LCD TV Sets. *IEEE Transactions on Electromagnetic Compatibility*, 55(3):564–570.
- Kumar, M., Garfinkel, T., Boneh, D., and Winograd, T. (2007). Reducing shoulder-surfing by using gaze-based password entry. In *Symposium on Usable Privacy and Security*.
- Lagorio, A., Tistarelli, M., Cadoni, M., Fookes, C., and Sridharan, S. (2013). Liveness detection based on 3d face shape analysis. In *Biometrics and Forensics (IWBF), International Workshop on*, pages 1–4.
- Langmead, B., Trapnell, C., Pop, M., Salzberg, S. L., et al. (2009). Ultrafast and memory-efficient alignment of short dna sequences to the human genome. *Genome Biol*, 10(3):R25.
- Laptev, I. and Pérez, P. (2007). Retrieving actions in movies. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–8. IEEE.
- Lavie, A. (2010). Evaluating the output of machine translation systems. *AMTA Tutorial*.
- Lavie, A. and Denkowski, M. J. (2009). The METEOR metric for automatic evaluation of machine translation. *Machine Translation*, 23(2-3):105–115.
- Law-To, J., Buisson, O., Gouet-Brunet, V., and Boujemaa, N. (2006). Robust voting algorithm based on labels of behavior for video copy detection. In *Proceedings of the 14th annual ACM international conference on Multimedia*, pages 835–844. ACM.
- Law-To, J., Buisson, O., Gouet-Brunet, V., and Boujemaa, N. (2009). Vicopt: a robust system for content-based video copy detection in large databases. *Multimedia systems*, 15(6):337–353.

- Law-To, J., Joly, A., and Boujemaa, N. (2007). Muscle-vcd-2007: a live benchmark for video copy detection.
- Lazar, J., Feng, J. H., and Hochheiser, H. (2010). *Research Methods in Human-Computer Interaction*. John Wiley and Sons.
- Lee, B. and Chun, J. (2009). Manipulation of virtual objects in marker-less AR system by fingertip tracking and hand gesture recognition. In *Proceedings of the 2nd International Conference on Interaction Sciences*.
- Lee, T. and Hollerer, T. (2007). Handy AR: Markerless inspection of augmented reality objects using fingertip tracking. In *IEEE International Symposium on Wearable Computers*.
- Li, Y., Li, Y., Yan, Q., Kong, H., and Deng, R. H. (2015). Seeing your face is not enough: An inertial sensor-based liveness detection for face authentication. In *Proceedings of the 22nd ACM Conference on Computer and Communications Security*, pages 1558–1569.
- Li, Y., Xu, K., Yan, Q., Li, Y., and Deng, R. H. (2014). Understanding osn-based facial disclosure against face authentication systems. In *Proceedings of the ACM Symposium on Information, Computer and Communications Security (ASIACCS)*, pages 413–424. ACM.
- Liao, W.-H. and Chang, C.-C. (2004). Embedding information within dynamic visual patterns. In *IEEE Multimedia and Expo*, volume 2, pages 895–898.
- Liu, J., Huang, Z., Cai, H., Shen, H. T., Ngo, C. W., and Wang, W. (2013). Near-duplicate video retrieval: Current research and future trends. *ACM Computing Surveys (CSUR)*, 45(4):44.
- Liu, Y., Gummadi, K. P., Krishnamurthy, B., and Mislove, A. (2011). Analyzing facebook privacy settings: user expectations vs. reality. In *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference*, pages 61–70. ACM.
- Lowry, R. (1998). *Concepts and Applications of Inferential Statistics*. Vassar College, <http://www.vassarstats.net/textbook/>.
- Lu, C. and Tang, X. (2014). Surpassing human-level face verification performance on lfw with gaussianface. *arXiv preprint arXiv:1404.3840*.
- Lucas, B. and Kanade, T. (1981). An iterative image registration technique with an application to stereo vision. In *Joint Conf. on Artificial Intelligence*, pages 674–679.
- Lutton, E., Maitre, H., and Lopez-Krahe, J. (1994). Contribution to the determination of vanishing points using hough transform. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 16(4):430–438.
- Määttä, J., Hadid, A., and Pietikainen, M. (2011). Face spoofing detection from single images using micro-texture analysis. In *Biometrics (IJCB), International Joint Conference on*, pages 1–7.
- Magee, M. and Aggarwal, J. (1984). Determining vanishing points from perspective images. *Computer Vision, Graphics, and Image Processing*, 26(2):256–267.

- Maggi, F., Volpatto, A., Gasparini, S., Boracchi, G., and Zanero, S. (2011). A fast eavesdropping attack against touchscreens. In *Information Assurance and Security (IAS)*. IEEE.
- Marquardt, P., Verma, A., Carter, H., and Traynor, P. (2011). (sp) iphone: decoding vibrations from nearby keyboards using mobile phone accelerometers. In *Proceedings of the 18th ACM conference on Computer and communications security*, pages 551–562. ACM.
- Marr, D. (1982). *Vision: a computational investigation into the human representation and processing of visual information*. W. H. Freeman, San Francisco.
- Marr, D. and Poggio, T. (1979). A computational theory of human stereo vision. *Proceedings of the Royal Society of London. Series B, Biological Sciences*, 204(1156):301–328.
- Marszałek, M., Laptev, I., and Schmid, C. (2009). Actions in context. In *IEEE Conference on Computer Vision & Pattern Recognition*.
- Mitra, N. J., Chu, H.-K., Lee, T.-Y., Wolf, L., Yeshurun, H., and Cohen-Or, D. (2009). Emerging images. *ACM Transactions on Graphics*, 28(5).
- Mo, F., Lu, Y.-H., Zhang, J.-L., Cui, Q., and Qiu, S. (2013). A support vector machine for identification of monitors based on their unintended electromagnetic emanation. *Progress In Electromagnetics Research M*, 30:211–224.
- Moon, Y.-S., Whang, K.-Y., and Han, W.-S. (2002). General match: a subsequence matching method in time-series databases based on generalized windows. In *Proceedings of the 2002 ACM SIGMOD international conference on Management of data*, pages 382–393. ACM.
- Mori, G. and Malik, J. (2003). Recognizing objects in adversarial clutter: breaking a visual CAPTCHA. In *Computer Vision and Pattern Recognition*, volume 1, pages 134–141.
- Motoyama, M., Levchenko, K., Kanich, C., McCoy, D., Voelker, G. M., and Savage, S. (2010). Re: CAPTCHAs-understanding CAPTCHA-solving services in an economic context. In *USENIX Security Symposium*, pages 435–462.
- Nakamura, J. (2005). *Image sensors and signal processing for digital still cameras*. CRC.
- Naor, M. (1996). Verification of a human in the loop or identification via the Turing test.
- Nguyen, D., Pham, T., and Jeon, J. (2009). Fingertip detection with morphology and geometric calculation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*.
- NuCaptcha (2011). Whitepaper: NuCaptcha & Traditional Captcha. <http://nucaptcha.com>.
- Oka, K., Sato, Y., and Koike, H. (2002). Real-time fingertip tracking and gesture recognition. *Computer Graphics and Applications*, 22(6):64–71.
- Oliva, A. and Torralba, A. (2007). The role of context in object recognition. *Trends in Cognitive Sciences*, 11(12):520 – 527.

- Owusu, E., Han, J., Das, S., Perrig, A., and Zhang, J. (2012). Accessory: password inference using accelerometers on smartphones. In *Proceedings of the Twelfth Workshop on Mobile Computing Systems & Applications*. ACM.
- Pantic, M., Pentland, A., Nijholt, A., and Huang, T. S. (2007). Human computing and machine understanding of human behavior: a survey. In *Artificial Intelligence for Human Computing*, pages 47–71. Springer.
- Parkhi, O. M., Vedaldi, A., and Zisserman, A. (2015). Deep face recognition. In *Proceedings of the British Machine Vision Conference (BMVC)*.
- Pavlovic, V. I., Sharma, R., and Huang, T. S. (1997). Visual interpretation of hand gestures for human-computer interaction: A review. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 19(7):677–695.
- Paysan, P., Knothe, R., Amberg, B., Romdhani, S., and Vetter, T. (2009). A 3d face model for pose and illumination invariant face recognition. In *Proceedings of the 6th IEEE International Conference on Advanced Video and Signal based Surveillance (AVSS) for Security, Safety and Monitoring in Smart Environments*.
- Peixoto, B., Michelassi, C., and Rocha, A. (2011). Face liveness detection under bad illumination conditions. In *Image Processing (ICIP), 18th IEEE International Conference on*, pages 3557–3560.
- Pérez, P., Gangnet, M., and Blake, A. (2003). Poisson image editing. *ACM Transactions on Graphics (TOG)*, 22(3):313–318.
- Polakis, I., Lancini, M., Kontaxis, G., Maggi, F., Ioannidis, S., Keromytis, A. D., and Zanero, S. (2012). All your face are belong to us: Breaking facebook’s social authentication. In *Proceedings of the 28th Annual Computer Security Applications Conference*, pages 399–408.
- Pouillot, S., Buisson, O., and Crucianu, M. (2007). Z-grid-based probabilistic retrieval for scaling up content-based copy detection. In *Proceedings of the 6th ACM international conference on Image and video retrieval*, pages 348–355. ACM.
- Qu, C., Monari, E., Schuchert, T., and Beyerer, J. (2014). Fast, robust and automatic 3d face model reconstruction from videos. In *Advanced Video and Signal Based Surveillance (AVSS), 11th IEEE International Conference on*, pages 113–118.
- Qu, C., Monari, E., Schuchert, T., and Beyerer, J. (2015a). Adaptive contour fitting for pose-invariant 3d face shape reconstruction. In *Proceedings of the British Machine Vision Conference (BMVC)*, pages 1–12.
- Qu, C., Monari, E., Schuchert, T., and Beyerer, J. (2015b). Realistic texture extraction for 3d face models robust to self-occlusion. In *IS&T/SPIE Electronic Imaging*. International Society for Optics and Photonics.

- Raffel, C. and Ellis, D. P. (2015). Large-scale content-based matching of midi and audio files. In *Proceedings of the 16th International Society for Music Information Retrieval Conference*, pages 234–240.
- Raguram, R., White, A., Goswami, D., Monroe, F., and Frahm, J. (2011). ispy: automatic reconstruction of typed input from compromising reflections. In *Proceedings of the 18th ACM conference on Computer and communications security*, pages 527–536. ACM.
- Raguram, R., White, A. M., Xu, Y., Frahm, J.-M., Georgel, P., and Monroe, F. (2013). On the privacy risks of virtual keyboards: automatic reconstruction of typed input from compromising reflections. *IEEE Transactions on Dependable and Secure Computing*.
- Ray, S. and Turi, R. (1999). Determination of number of clusters in k-means clustering and application in colour image segmentation. In *Proceedings of the International conference on advances in pattern recognition and digital techniques*, pages 137–143.
- Rodriguez, M. D., Ahmed, J., and Shah, M. (2008). Action mach a spatio-temporal maximum average correlation height filter for action recognition. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE.
- Samal, A. and Iyengar, P. A. (1992). Automatic recognition and analysis of human faces and facial expressions: A survey. *Pattern recognition*, 25(1):65–77.
- Schops, T., Sattler, T., Hane, C., and Pollefeys, M. (2015). 3d modeling on the go: Interactive 3d reconstruction of large-scale scenes on mobile devices. In *3D Vision (3DV), International Conference on*, pages 291–299.
- Schroff, F., Kalenichenko, D., and Philbin, J. (2015). Facenet: A unified embedding for face recognition and clustering. *arXiv preprint arXiv:1503.03832*.
- Seo, H. J. and Milanfar, P. (2009). Detection of human actions from a single example. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 1965–1970. IEEE.
- Shen, H. T., Ooi, B. C., and Zhou, X. (2005). Towards effective indexing for very large video sequence database. In *Proceedings of the ACM SIGMOD international conference on Management of data*, pages 730–741. ACM.
- Shi, F., Wu, H.-T., Tong, X., and Chai, J. (2014). Automatic acquisition of high-fidelity facial performances using monocular videos. *ACM Transactions on Graphics (TOG)*, 33(6):222.
- Shirali-Shahreza, M. and Shirali-Shahreza, S. (2008). Motion CAPTCHA. In *Conference on Human System Interactions*, pages 1042–1044.
- Shukla, D., Kumar, R., Serwadda, A., and Phoha, V. V. (2014). Beware, your hands reveal your secrets! In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pages 904–917. ACM.
- Simard, P., Steinkraus, D., and Platt, J. (2003). Best practices for convolutional neural networks applied to visual document analysis. In *Proceedings of the Seventh International Conference on Document Analysis and Recognition*, volume 2, pages 958–962.

- Smeaton, A. F., Over, P., and Kraaij, W. (2006). Evaluation campaigns and trecvid. In *Proceedings of the 8th ACM international workshop on Multimedia information retrieval*, pages 321–330. ACM.
- Sobrado, L. and Birget, J.-C. (2002). Graphical passwords. *The Rutgers Scholar*, 4.
- Soomro, K. and Zamir, A. R. (2014). Action recognition in realistic sports videos. In *Computer Vision in Sports*, pages 181–208. Springer.
- Soundararajan, R. and Bovik, A. C. (2013). Video quality assessment by reduced reference spatio-temporal entropic differencing. *Circuits and Systems for Video Technology, IEEE Transactions on*, 23(4):684–694.
- Soupionis, Y. and Gritzalis, D. (2010). Audio CAPTCHA: Existing solutions assessment and a new implementation for VoIP telephony. *Computers & Security*, 29(5):603–618.
- Stelzer, E. (1998). Contrast, resolution, pixelation, dynamic range and signal-to-noise ratio: fundamental limits to resolution in fluorescence light microscopy. *Journal of Microscopy*, 189(1):15–24.
- Stevenage, S. V., Nixon, M. S., and Vince, K. (1999). Visual analysis of gait as a cue to identity. *Applied cognitive psychology*, 13(6):513–526.
- Sun, L., Pan, G., Wu, Z., and Lao, S. (2007). Blinking-based live face detection using conditional random fields. In *Advances in Biometrics*, pages 252–260. Springer.
- Sun, Y., Wang, X., and Tang, X. (2013). Deep convolutional network cascade for facial point detection. In *Computer Vision and Pattern Recognition (CVPR), IEEE Conference on*, pages 3476–3483.
- Suwajanakorn, S., Kemelmacher-Shlizerman, I., and Seitz, S. M. (2014). Total moving face reconstruction. In *Computer Vision–ECCV 2014*, pages 796–812. Springer.
- Suwajanakorn, S., Seitz, S. M., and Kemelmacher-Shlizerman, I. (2015). What makes tom hanks look like tom hanks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3952–3960.
- Szeliski, R. (2006). Image alignment and stitching: A tutorial. *Foundations and Trends® in Computer Graphics and Vision*, 2(1):1–104.
- Taigman, Y., Yang, M., Ranzato, M., and Wolf, L. (2014). Deepface: Closing the gap to human-level performance in face verification. In *Computer Vision and Pattern Recognition (CVPR), IEEE Conference on*, pages 1701–1708.
- Tan, D. S., Keyani, P., and Czerwinski, M. (2005). Spy-resistant keyboard: More secure password entry on public touch screen displays. In *Proceedings of the 17th Australia Conference on Computer-Human Interaction*.

- Tan, H.-K., Ngo, C.-W., Hong, R., and Chua, T.-S. (2009). Scalable detection of partial near-duplicate videos by visual-temporal consistency. In *Proceedings of the 17th ACM international conference on Multimedia*, pages 145–154. ACM.
- Tan, X., Li, Y., Liu, J., and Jiang, L. (2010). Face liveness detection from a single image with sparse low rank bilinear discriminative model. In *European Conference on Computer Vision (ECCV)*, pages 504–517.
- Tanskanen, P., Kolev, K., Meier, L., Camposeco, F., Saurer, O., and Pollefeys, M. (2013). Live metric 3d reconstruction on mobile phones. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 65–72.
- Thorpe, S., Fize, D., and Marlot, C. (1996). Speed of processing in the human visual system. *Nature*, 381(6582):520–522.
- Torralba, A. and Freeman, W. T. (2012). Accidental pinhole and pinspeck cameras: revealing the scene outside the picture. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 374–381. IEEE.
- Tsin, Y., Ramesh, V., and Kanade, T. (2001). Statistical calibration of ccd imaging process. In *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, volume 1, pages 480–487. IEEE.
- Turk, M. A. and Pentland, A. P. (1991). Face recognition using eigenfaces. In *Computer Vision and Pattern Recognition, 1991. Proceedings CVPR’91., IEEE Computer Society Conference on*, pages 586–591. IEEE.
- Ukita, N. and Kidode, M. (2004). Wearable virtual tablet: fingertip drawing on a portable plane-object using an active-infrared camera. In *Proceedings of the International Conference on Intelligent User Interfaces*. ACM.
- Ullman, S. (1983). Computational studies in the interpretation of structure and motion: Summary and extension. In *Human and Machine Vision*. Academic Press.
- Ullman, S. (2000). *High-Level Vision: Object Recognition and Visual Cognition*. The MIT Press.
- Valentin, D., Abdi, H., O’Toole, A. J., and Cottrell, G. W. (1994). Connectionist models of face processing: A survey. *Pattern recognition*, 27(9):1209–1230.
- van Eck, W. (1985). Electromagnetic radiation from video display units: an eavesdropping risk? *Computer Security*, 4:269–286.
- Vedaldi, A. and Fulkerson, B. (2010). VLFeat: An open and portable library of computer vision algorithms. In *Intl. conference on Multimedia*, pages 1469–1472.
- Ventura, J., Arth, C., Reitmayr, G., and Schmalstieg, D. (2014). Global localization from monocular slam on a mobile phone. *Visualization and Computer Graphics, IEEE Transactions on*, 20(4):531–539.

- Viola, P. A. and Jones, M. J. (2001). Rapid object detection using a boosted cascade of simple features. In *Computer Vision and Pattern Recognition*.
- Von Ahn, L., Blum, M., Hopper, N. J., and Langford, J. (2003). Captcha: Using hard ai problems for security. In *Advances in CryptologyEUROCRYPT 2003*, pages 294–311. Springer.
- von Ahn, L., Blum, M., and Langford, J. (2004). Telling humans and computers apart automatically. *Commun. ACM*, 47:56–60.
- Vuagnoux, M. and Pasini, S. (2009). Compromising electromagnetic emanations of wired and wireless keyboards. In *Proceedings of the 18th USENIX Security Symposium*.
- Wang, L., Hu, W., and Tan, T. (2003). Recent developments in human motion analysis. *Pattern recognition*, 36(3):585–601.
- Wang, T., Yang, J., Lei, Z., Liao, S., and Li, S. Z. (2013). Face liveness detection using 3d structure recovered from a single camera. In *Biometrics (ICB), International Conference on*, pages 1–6.
- Weaver, J., Mock, K. J., and Hoanca, B. (2011). Gaze-based password authentication through automatic clustering of gaze points. In *IEEE International Conference on Systems, Man and Cybernetics*.
- Wei, S., Zhao, Y., Zhu, C., Xu, C., and Zhu, Z. (2011). Frame fusion for video copy detection. *Circuits and Systems for Video Technology, IEEE Transactions on*, 21(1):15–28.
- Weickert, J. (1998). *Anisotropic diffusion in image processing*, volume 1. Teubner Stuttgart.
- Widrow, B. and Kollár, I. (2008). *Quantization noise: roundoff error in digital computation, signal processing, control, and communications*. Cambridge University Press.
- Willems, G., Tuytelaars, T., and Van Gool, L. (2008). Spatio-temporal features for robust content-based video copy detection. In *Proceedings of the 1st ACM Int. Conf. on Multimedia information retrieval*, pages 283–290. ACM.
- Wu, H.-Y., Rubinstein, M., Shih, E., Gutttag, J., Durand, F., and Freeman, W. T. (2012). Eulerian video magnification for revealing subtle changes in the world. *ACM Transactions on Graphics (TOG)*, 31(4).
- Wu, X., Hauptmann, A. G., and Ngo, C.-W. (2007). Practical elimination of near-duplicates from web video search. In *Proceedings of the 15th international conference on Multimedia*, pages 218–227. ACM.
- Xiong, X. and De la Torre, F. (2013). Supervised descent method and its applications to face alignment. In *Computer Vision and Pattern Recognition (CVPR), IEEE Conference on*, pages 532–539.
- Xu, Y., Frahm, J.-M., and Monroe, F. (2014a). Watching the watchers: Automatically inferring tv content from outdoor light effusions. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, Scottsdale, AZ*, pages 418–428. ACM.

- Xu, Y., Heinly, J., White, A. M., Monroe, F., and Frahm, J.-M. (2013). Seeing double: reconstructing obscured typed input from repeated compromising reflections. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security, Berlin, Germany*, pages 1063–1074. ACM.
- Xu, Y., Reynaga, G., Chiasson, S., Frahm, J.-M., Monroe, F., and van Oorschot, P. C. (2012). Security and usability challenges of moving-object captchas: Decoding codewords in motion. In *USENIX Security Symposium, Bellevue, WA*, pages 49–64.
- Xu, Y., Reynaga, G., Chiasson, S., Frahm, J.-M., Monroe, F., and van Oorschot, P. C. (2014b). Security analysis and related usability of motion-based captchas: Decoding codewords in motion. *Dependable and Secure Computing, IEEE Transactions on*, 11(5):480–493.
- Yan, J. and Ahmad, A. S. E. (2007). Breaking visual CAPTCHAs with naive pattern recognition algorithms. In *ACSAC*, pages 279–291.
- Yan, J. and Ahmad, A. S. E. (2008a). A low-cost attack on a Microsoft CAPTCHA. In *ACM Conference on Computer and Communications Security*, pages 543–554.
- Yan, J. and Ahmad, A. S. E. (2008b). Usability of CAPTCHAs or usability issues in CAPTCHA design. In *SOUPS*, pages 44–52.
- Yan, J. and El Ahmad, A. (2011). CAPTCHA robustness: A security engineering perspective. *Computer*, 44(2):54–60.
- Yan, J. and Pollefeys, M. (2007). Articulated motion segmentation using RANSAC with priors. *Dynamical Vision*, pages 75–85.
- Yang, D., Jin, L., and Yin, J. (2005). An effective robust fingertip detection method for finger writing character recognition system. In *Proceedings of the International Conference on Machine Learning and Cybernetics*.
- Yang, H. and Meinel, C. (2014). Content based lecture video retrieval using speech and video text information. *Learning Technologies, IEEE Transactions on*, 7(2):142–154.
- Yang, J., Lei, Z., Liao, S., and Li, S. Z. (2013). Face liveness detection with component dependent descriptor. In *Biometrics (ICB), International Conference on*, pages 1–6.
- Yeh, M.-C. and Cheng, K.-T. (2009). A compact, effective descriptor for video copy detection. In *Proceedings of the 17th ACM international conference on Multimedia*, pages 633–636. ACM.
- Yilmaz, A., Javed, O., and Shah, M. (2006). Object tracking: A survey. *ACM Comput. Surv.*, 38.
- Youtube (2015). Statistics from youtube website. <https://www.youtube.com/yt/press/statistics.html>. Accessed: 2015-04-28.
- Yuan, J., Duan, L.-Y., Tian, Q., and Xu, C. (2004). Fast and robust short video clip search using an index structure. In *Proceedings of the ACM SIGMM international workshop on Multimedia information retrieval*, pages 61–68. ACM.

- Yue, Q., Ling, Z., Liu, B., Fu, X., and Zhao, W. (2014). Blind recognition of touched keys: Attack and countermeasures. *arXiv preprint arXiv:1403.4829*.
- Zhang, D.-Q. and Chang, S.-F. (2004). Detecting image near-duplicate by stochastic attributed relational graph matching with learning. In *Proceedings of the 12th annual ACM international conference on Multimedia*, pages 877–884. ACM.
- Zhang, L., Curless, B., and Seitz, S. M. (2002). Rapid shape acquisition using color structured light and multi-pass dynamic programming. In *3D Data Processing Visualization and Transmission, First International Symposium on*, pages 24–36.
- Zhang, L. and Rui, Y. (2013). Image search—from thousands to billions in 20 years. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMCCAP)*, 9(1s):36.
- Zhang, L. and Samaras, D. (2006). Face recognition from a single training image under arbitrary unknown lighting using spherical harmonics. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 28(3):351–363.
- Zhang, Y., Xia, P., Luo, J., Ling, Z., Liu, B., and Fu, X. (2012). Fingerprint attack against touch-enabled devices. In *Security and Privacy in Smartphones and Mobile Devices, SPSM '12*.
- Zhang, Z. (2003). Vision-based interaction with fingers and papers. In *Proceedings International Symposium on the CREST Digital Archiving Project*.
- Zhu, B. B., Yan, J., Li, Q., Yang, C., Liu, J., Xu, N., Yi, M., and Cai, K. (2010). Attacks and design of image recognition CAPTCHAs. In *ACM Conference on Computer and Communications Security*, pages 187–200.
- Zhu, X., Lei, Z., Yan, J., Yi, D., and Li, S. Z. (2015). High-fidelity pose and expression normalization for face recognition in the wild. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 787–796.
- Zhuang, L., Zhou, F., and Tygar, J. (2005). Keyboard acoustic emanations revisited. In *Proceedings of the 12th ACM Conference on Computer and Communications Security*.